





Digitized by the Internet Archive  
in 2025

<https://archive.org/details/openhousesapporo02tedn>



## IntelligentPad:

### ミームメディアとミームプールのアーキテクチャ

田中 譲

北海道大学工学部電気工学科

本稿では、IntelligentPadの開発を通じて、著者のIntelligentPadに対する意味付けがどのように変化してきたかを、メディア・ツールキット・システムからシンセティック・メディア・システム、さらにはミームメディア・システムへの変遷を追って述べ、IntelligentPadがどのように機能強化されてきたか、システムと設計者の共進化の過程を振り返る。さらに、ミームメディアにとって重要な、ネットワークを介した流通と市場(いちば)システムの提供に関して、WWWとNetscapeを用いたワールド・ワイドな市場システムの構築と、IntelligentPadをInternetのフロントエンドとした市場システムの構築、さらには、HTMLビューワのIntelligentPad部品としての実現について概説する。

## IntelligentPad:

### Architectures for Meme Media and a Meme Pool

Yuzuru TANAKA

Electrical Engineering Department, Hokkaido University

The IntelligentPad system was first developed as a media toolkit system, then it was improved to become a synthetic media system, and is now aiming to become a meme media. It has been coevolving together with its architect's understanding of the system. This paper first describes this coevolution process, and in the latter half it describes how the IntelligentPad system can provide new media called pads for the world-wide distribution of various intellectual resources through the Internet. It proposes three systems. The first one uses Netscape as a catalog of pads to form a world-wide market place of pads. The second one uses the IntelligentPad itself as a frontend of the Internet to form a functionally enriched world-wide market place of pads. The third one provides an HTML viewer as a pad on which any composite pad, even another HTML viewer pad, can be pasted to distribute itself together with the content HTML sentences. The third one seamlessly integrates HTML browsers with the pad distribution system.

#### 1 はじまり

IntelligentPadシステムの開発プロジェクトは1987年初頭に著者によって始められた。当時はパーソナル・コンピュータの普及が急速に進んだ時期である。パーソナル・コンピュータの一般ユーザへの普及は、パーソナル・コンピュータの主たる利用形態が、自身でプログラムを開発して利用するという形態から、市販のツールを用いて文書作成や数値処理に利用するという形態へと転換したことを物語っていた。ワード・プロセッサ、スプレッド・シート、チャート・ツール、ドローイング・ツールなどのツール・ソフトウェアを用いた利用が一般ユーザによるパーソナル・コンピュータ利用の殆どを占めていた。さらには、これらのツール間での連携機能の提供も行われるようになっていた。スプレッド・シートを用いて計算した数表をチャート・ツールを用いてグラフ表示し、このグラフをワードプロセッサを用いて作成した文書中に埋めこむというようなことが可能な、統合ビジネス・ソフトウェア・システムが市場に出るようになってきていた。しかし、いずれの統合ビジネス・ソフトウェア・システムも、基本ツールはワード・プロセッサ、スプレッド・シート、チャート・ツール、ドローイング・ツールなどのごく少数のツールに限られており、利用者が新しく別のツールを付加することは不可能であり、ツール間の連携も機能と表現が予め想定されている定形業務に対応するものに限られていた。

利用者自身が種々の機能を取り出し自分が必要とする処理と表現を実現できるように組み合わせて連携させることは絶望的に困難であったり、本質的に不可能である。このことこそが、利用者が日常最も悩まされている典型的な問題である。この問題を解決するには、種々の機能をエンド・ユーザが直接取り扱え、それらの間の自在な連携をエンド・ユーザが直接定義できることが必要である。ところで、エンド・ユーザが直接構成可能であることが望まれるシステムとは、決してDBMSやコンパイラ、あるいは数値計算プログラムのようなサーバシステムではない。これらのサーバシステムに対するクライアントシステムこそが、エンド・ユーザの多様なニーズに答えるべく、エンド・ユーザ自身による臨機応変な構成を必要とするのである。クライアント・システムは単なるGUIではない。エディタはクライアント・システムであるが、エディタはGUIのみでは構成できない。エディタは、テキストや数表や図面を表示する一枚の紙の形状をしたウィンドウというディスプレイ・オブジェクトと、それに対するユーザの直接操作が発するユーザ・イベントに応じて文書編集処理や数表計算処理を行う内部プログラムとからなっている。このように一枚の紙の形状をしたディスプレイ・オブジェクトと内部プログラムとが一体化したオブジェクトを、GUIオブジェクトと区別するために、著者はメディア・オブジェクトと名付けた。

#### 2. メディア・ツールキットとしてのIntelligentPad







当時の著者の目論見は、百種類余りのメディア・オブジェクトを用意し、エンドユーザにこれらの機能連携の自在な定義を許すことにより、マルチメディア文書と各種のツールをエンドユーザが自在に合成できるシステムを開発することにあった。そのために必要なメディア・オブジェクトの種類は百種類余りで十分であろうと漠然と考え、必要にして十分なコンストラクション・セットを明らかにしそれを提供することを目指していた。メディア・オブジェクト間の連携は、紙に紙を貼るように、ディスプレイ上でメディア・オブジェクトにメディア・オブジェクトを貼る埋め込みによる連携により定義する方法を採用した。

著者はマルチメディア文書と各種ツールを同様に扱うことにこだわり、両者の部品を統一的にパッドと呼ばれるメディア・オブジェクトによって表現することにした。メディア・オブジェクトの連携もマルチメディア文書の編集と同様の操作で定義可能にすることにこだわり、埋め込みによる連携定義法を採用した。この頃のシステムを著者はメディア・ツールキット・システムと呼んでいた。

メディア・オブジェクトの連携に埋め込みによる連携定義法を採用したメディア・ツールキット・システムは、マルチメディア複合文書の自然な拡張とも見なせる。1980年代のデスクトップ・パブリッシング(DTP)やマルチメディア・オーサリングの発展はメディア編集の技術と文化を著しく発展させた。その発展過程は概ね以下のように進行した。DTPではテキスト中に図表や図面、画像を自在に配置した複合文書の作成と印刷が可能となった。TeXやPostScriptなどの印刷フォントやフォーマットを制御する言語が開発され、ディスプレイ上でも印刷形式で表示できるようになり、WYSIWYG(What You See Is What You Get)の設計思想が広まった。さらに、印刷形式でアニメーションを記述できるDisplay PostScriptが開発されたことにより、複合文書中にアニメーションを埋め込んで、これを電子メールで送るといったようなことも可能になった。さらに、ビデオ映像を他のウィンドウと混在させて表示する機能を持ったビデオ・ボードが各社から発表され、複合文書中に音声やビデオクリップなども埋め込みプレゼンテーションに用いる試みも現れてきた。DTPR(Desk Top Presentation)システムの台頭である。

次に研究者らは、異なるタイプの要素を統一的に取り扱う方法を捜し求めた。そしてオブジェクト指向のシステム開発法が導入され、各種のマルチメディア・オブジェクトのみならずアプリケーション・プログラムも一様にオブジェクトとして取り扱えるようになった。これにより、プログラムも複合文書の要素として埋め込むことが可能になった。まず、スクリプト・プログラムが定義されたコマンド・ボタンが導入された。文書中に埋め込まれたコマンド・ボタンをクリックするとスクリプト・プログラムが実行される。このような試みは、複合文書が、その要素として、ユーザ操作に反応して状態変化をおこす要素も持つように一般化されていった。このような複合文書をリアクティブ・ドキュメントとかダイナミック・メディア・ドキュメントという。リアクティブ・ドキュメントでは、文書中の数値の数値を書き換えると対応するチャートが自動的に書き換えられる。

80年代中頃に、オブジェクト指向の合成プログラミング・パラダイムがウィンドウ・システムの種々のGUIツールキットを生んだ。このような背景を受けて、リアクティブ・ドキュメントにツールキット化を適用したものがメディア・ツールキット・システムIntelligentPadである。

### 3. シンセティック・メディアとしてのIntelligentPad

このような目論見で著者のグループがメディア・ツールキット・システムとしてのIntelligentPadの開発を進める内に、メディア・オブジェクト部品の種類が百種類余りで十分であろうという考えが大きな誤りであることが明らかになってきた。1989年頃のことである。そもそも、メディア・オブジェクト部品の種類に上限を設けることができないことがその頃初めて認識された。さらに、マルチメディア文

書の構成要素や各種ツールが統一的にパッドによって表現されるのみならず、DBMSや電子メールなどの各種サーバシステム、あらゆるアプリケーション・システム、さらにはそれらを統合的に用いるための統合環境を与える場までもが、パッドとして統一的に表現され得ることに気付くに至った。それらの多くは、白紙のパッドとして表現される。意味のあるGUIインタフェースを持たないオブジェクトに対しても白紙のパッドによる表現を与えることにより、埋め込みによる連携定義を可能にしたのである(図1)。

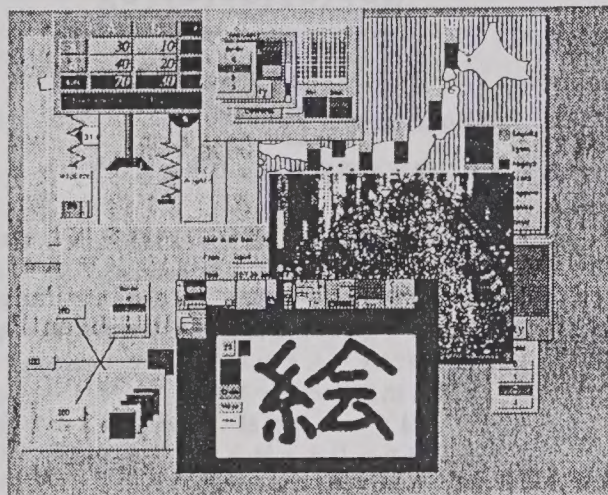


図1. IntelligentPadのディスプレイ

もっともこのことは当初からある程度は予想されていたことである。「パッド」という、イメージを狭く特定することのない名前を選んだのはこのためである。このような拡張により、多様な部品パッドを、連携相手を予め想定しないジェネリックなパッドとして定義しておくことが重要になり、パッド間の連携のためのメッセージ交換インタフェースの標準化が必要となった。その結果、パッドの論理的構造をスロット・リストとし、パッド間の連携をset、gimme、updateの3種類の標準メッセージを用いたスロット結合により実現する現在のアーキテクチャが確立された。著者はこのようなアーキテクチャをシンセティック・メディア・アーキテクチャと名付けた。

### 4. ミーム・メディアとしてのIntelligentPadへの発展

パッドの種類に制限がなく、多様なツールをパッドとして実現できるとすると、誰がそのような多様なパッドを提供することになるのだろうか?基本部品パッドの開発はプロの仕事であるから、企業や大学のソフトウェア開発者によって開発されることになる。開発グループは一社に特定できるであろうか?そのようなことをすれば、IntelligentPadと同じようなアーキテクチャのシステムが複数の会社から製品化され、それぞれが相互に連携することができない互換性のない部品を提供することになる。それでは結局は現在の状況と同じ問題を引き起こす。同じような機能が何度も繰り返し開発され、いろいろなシステム中に存在しているにもかかわらず、ユーザがこの機能を、自身が必要としている環境で、自身が必要とする部品と連携させて使おうとすると、原理的に不可能であったり、絶望的な困難に遭遇することになる。この問題を解決するには、多数の部品パッド開発会社が、相互の部品の連携を保証しつつ多様な部品パッドを提供し得るようになり、IntelligentPadのカーネル・システムを共通社会基盤にまで発展させる必要がある。

このような考えから、1991年頃、著者の関心は情報の生態系を考慮したシステム・アーキテクチャへと広がる。IntelligentPadを部品パッド提供会社のための共通社会基盤へと発展させるだけでなく、あらゆる知的資源の「編集」と「流通」と「管理」の基盤アーキテクチャとして見直す見方が次第に育っていった。部品パッドの提供会社だけで







なく、パッドの利用者であり、編集者ともなりうるエンドユーザをも含めた社会の基盤アーキテクチャとして見直したのである。こうして行き着いた所が1993年より提唱しているミーム・メディアの概念である。ミーム・メディアは社会において流通され編集されるあらゆる知的資源からなる文化の発展を支援するメディアである。

文化とは、外在化された知識や情報が共有され、引用され編集されて新しい知識や情報を生み、それらが順に積み上げられたものと考えることができる。知識や情報の外在化によって、個体が獲得した知識や情報が体外に記録・保存され、コミュニティにおいて流通するようになり、共有されるようになる。そうすると、保存されている情報の引用や再利用が可能になり、それを用いて新しい知識や情報を編集することが可能になる。新しい知識や情報は同様に共有され、コミュニティが保存する共有知識の上に積み上げられる。これが文化といわれるものである。この過程で、我々は情報の(1)外在化、(2)記録・保存、(3)流通、(4)共有、(5)引用、(6)編集の6つの情報活動を行っている。社会において流通され編集されるあらゆる知的資源からなる文化の発展を支援するには、これら6つの情報活動をすべて統合的に支援できる新しいメディアをコンピュータ上に定義し、そのメディア文化を育てることが必要である。このようなメディアは、編集と流通の機能を持つことが特徴である。

社会における文化の編集と流通の媒体としてのメディアは、遺伝子と同様に編集、複製が可能で、社会によって自然淘汰されるという特性を持ち、R. ドーキンスのいうミーム(文化遺伝子)の特性を持つミーム・メディアであるといえる。遺伝子は、組み替えと、増殖と、淘汰によって進化してきた。R. ドーキンスは同様の機構が文化の進化にも働いているのではないかと考え、ミーム(文化遺伝子、meme)という概念を提案した(1)。組み替えは編集に、増殖は複製と流通に、淘汰は引用に対応する。

知識の流通のためのメディアの開発の必要性は、1986年にXerox PARCのM. Stefikによって指摘されている。彼もまたミームについて触れ、ミームのような性質を持ったメディアの必要性を論じている(2)。

## 5. IntelligentPadアーキテクチャ

知的資源の編集・流通・管理は外在化された知的資源に対して行われる。知的資源の外在化はメディアを用いて行われる。編集・流通・管理のすべての対象となりうる単一種類のメディアが必要である。このメディアは、知的資源の部品化が可能で、部品の機能連携が部品の組合せ編集によって定義できなくてはならない。

各部品をジェネリックに定義するには、どのような相手と機能連携するかを予め仮定する訳にはいかない。したがって、部品のどのような機能に連結するのかを編集時に簡便に指示できるようなアーキテクチャが望ましい。そこで、IntelligentPadでは、丁度AV機器がコンポネント化されたときに、各機器のインタフェースが複数個のコネクション・ジャックの装備という形を取ったように、各部品の外部インタフェースをコネクション・ジャックの役目を果たすスロットを複数個定義することでモデル化する(図2)。これをスロット・リストと呼ぶ。さらに、各部品を別の部品のジャック、つまりスロットに結合できるように、ピン・プラグが一本用意されている。ピン・プラグを一本に制限することにより、結合構造を単純化し、各部品のモジュラリティを高め、合成パッドの再編集が容易になることをねらっている。

各スロットのアクセス・メッセージはset <slot\_name> <value> とgimme <slot\_name> の2種類に限定した。各スロットには、setとgimmeに対する手続が定義される。さらに、部品Aを部品Bに貼るときには、AのBに対する結合として、Bのスロットの内のただ一つを選ぶことができ、Aはこの選ばれたスロットに対してのみsetないしはgimmeメッセージ、あるいは両方を送ることができる。A

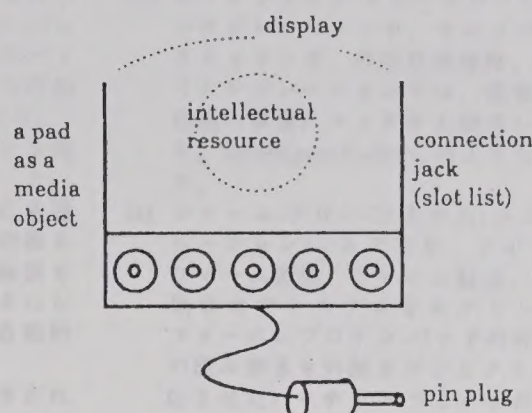


図2. パッドのジェネリックな定義を可能にする論理構造

のプログラミングにおいては、結合相手のマスター・パッドに対してsetとgimmeを発する場合、スロット名をパラメータに指定しないでおく。プログラムの実行の際には、貼り合わせの際に指定されたマスター・パッドのスロットの名前がsetやgimmeメッセージのパラメータに付加されてこれらのメッセージがマスター・パッドへと送出される。これにより、各パッドの動作を、貼付によって結合される相手のパッドに依存することなくジェネリックに定義することができる。

マスター・パッドのBからAには、パラメータなしのupdateメッセージを送ることができる。Bがupdateをどう用いるかは、Bのプログラムの責任に任される。送られてきたupdateに対してAがどう反応するかもまた、Aのプログラムの責任に任される。

貼り合わされた2枚のパッド間に許されるメッセージには、set、gimme、updateの3種類のメッセージの外にも、move、copy、resizeなど、すべてのパッドに共通の属性を変更するメッセージなどのように、いずれのパッドにも送ることができるいくつかのメッセージが含まれる。

貼りつけるパッドをマスター・パッドの適切なスロットに対応付けるには、サブパッドをマスター・パッドの上に移動したのち、コマンド・メニューを開いてconnectionを選び、コネクション・シートを開いて指定する。サブパッドにはこのスロット名が保持される。

新しいパッドは、プログラムで定義するか、既にあるパッドを貼り合わせて合成することができる。パッドは自在にコピーすることができる。コピーによってのみ、新しいパッドが生成される。パッドはすべて永続性を持つ。陽に消去しない限り、パッドは存在し続ける。IntelligentPadでは、プリミティブ・パッドも合成パッドも、どのパッドに対しても常に共有コピーを即座にとることができる。

## 6. IntelligentPadの適用分野

IntelligentPadは工夫しだいで、広範な分野に適用することができる。その主なものを以下に列挙する。

### (1) システム開発ツール

- (a) GUIツールキット: IntelligentPadはGUIツールキットの機能を含む。
- (b) クライアント・システム開発ツール: クライアント・システムの部品化を図ることにより、直接合成によるプログラム開発が可能になる。顧客のオフィスで、顧客のニーズを反映しながらインタフェース設計を行うと同時にシステム開発を済ませてしまうことが可能になる。
- (c) プラントや外部装置のコンソール・パネル設計: プラントや外部装置に対して、これらとのインタフェース・プログラムに、パッドとしての基本構造とスロット・リストを持たせたパッドを定義することが可能である。この







ように、外部オブジェクトとのインタフェースを行うパッドをプロキシ(proxy)パッドという。プロキシパッドは、パッドとして他のパッドに貼ったり、他のパッドをこの上に貼ることができるので、プラントや外部装置のコンソール・パネル設計に用いることができる。

- (d) 地理情報システム(GIS): パッドはバックグラウンドを透明にして何枚も重畳して貼ることができるので、コンピュータ・マッピングには適している。マップ上に各種のパッドを配置し、そのクリックによって詳細情報を表示させたり、それらに常にデータベースへの検索をさせ、時々刻々変化する情報を表示させたり、それらの配置位置自体をデータベースの更新に従って自動的に変化させることも可能である。
- (e) CAD: 特に、レイアウト系のCADへの応用が期待される。
- (f) システム・インテグレーション(ビュー統合): ビュー統合というのは、もともと統合されていないいくつかの異種システム(たとえばデータベースなど)が管理するリソースをディスプレイ上で関連づけて統合表示する手法である。異種システムに対し要求を出す際、これらの要求のパラメータの間に関連性を定義しておき、得られた処理結果を共通台紙の上にレイアウトして表示することによって、ディスプレイ上でこれらの結果を統合することができる。

## (2) パーソナル・オーグメンテーション

- (g) マルチメディア・オーサリング: IntelligentPadはマルチメディア・オーサリングのパラダイムを含むシステムである。もちろん、種々の新しいマルチメディア処理用パッドの開発が今後も必要である。
- (h) ハイパーテキスト/ハイパーメディア: ハイパーテキスト/ハイパーメディアにおけるアンカーとリンクもパッドとして開発されている。テキスト中の任意の文字列をアンカーに指定するためには、これをパッドとみなす疑似パッドの機構が必要となる。
- (i) データベースのフォーム・コンストラクション・キット: DBMSに対してもプロキシパッドを定義することができるので、これに他のパッドを自在に貼ることにより、インタフェース・フォームを自在に設計することができる。IntelligentPadのすべてのパッドがフォーム・コンストラクション・キットを構成する。
- (j) CAI: 各種のマイクロワールドを定義するのに適している。釧路工業高等専門学校の野口孝文助教授が数多くの試みを行っている。
- (k) 発想支援: 各種の発想支援ツールの開発が可能である。

## (3) グループ・オーグメンテーション

- (l) 共有スクリーン: 共有コピーの機能と、フィールド・パッドを用いて、任意のパッドを共有スクリーンにのせて利用することができる。
- (m) ビデオ会議: ビデオパッドがすでに開発されているので、ビデオ会議システムの開発は容易である。さらに、各種のパッドを貼ることにより、各種の機能を加えることが可能である。
- (n) 議論支援: gIBISのようなシステムもマルチメディアに拡張して容易に実現できる。
- (o) マルチメディア・メール/マルチメディアBBS: マルチメディア・メールは既に開発されている。マルチメディアBBSの開発も容易である。
- (p) ワーク・フロー支援: ワーク・フロー支援は北海道大学で試みられている。
- (q) グループ発想支援: 各種のグループ発想支援ツールの開発が可能である。

## (4) オーガナイズेशन・オーグメンテーション

- (r) エンタプライズ・インテグレーション: ボトム・アップ・インテグレーションや、リエンジニアリング、リストラクチャリング、非定形業務等、今後のエンタプライズ・インテグレーションでは、柔軟かつ速やかに変化し、現場の事情にマッチする統合システムの開発が望まれる。IntelligentPadはこのようなニーズにマッチしている。
- (s) フォーム・フロー・システム: エンタプライズ・インテグレーション・システムを、フォームの流れととらえ、フォーム変換、フォーム転送、トリガーの基本要素の組合せでシステムをモデリングする方法である。フォームにプロキシパッドの考えを導入し、フォームの読み書きを外部オブジェクトのモニターと制御に対応させたバーチャル・フォームの導入により、組織におけるすべての処理をフォーム・フローを介して制御することができる。

## (5) ソーシャル・オーグメンテーション

- (t) マルチメディア電子図書館: Internetを介したマルチメディア電子図書館は、ワールド・ワイド・ウェブおよびMosaic、NetscapeとIntelligentPadを結合することにより、容易に開発できる。
- (u) マルチメディア新聞: 上記と同様である。
- (v) ヴァーチャル・モール: ワールド・ワイド・ウェブのクライアントにMosaicやNetscapeの代わりにIntelligentPadを用いることができれば、ビデオパッドを用いてインタラクティブ・ビデオでヴァーチャル・モールを定義し、そのパッド上に各種パッドを貼ることにより、ショッピングを可能にすることができる。もちろん、購入トランザクションの処理に関してはシステムの開発が必要であり、これを利用するための特殊パッドも開発されねばならない。
- (w) ビデオ・オン・デマンドのインタラクティブ・プログラム案内: ビデオ・オン・デマンドやケーブルTVのインタラクティブ・プログラム・ガイドが米国ではビッグ・ビジネスになると考えられているが、このようなシステムの開発にもIntelligentPadが適している。
- (x) 科学技術に関する知的資源の出版・流通システム: 科学技術論文の出版にもパッドを用いることにより、例えば数式には裏にその数式を評価するプログラムを忍ばせたり、チャートには生データのテーブルを忍ばせて、これらを複合文書に埋めこんで出版することがネットワーク上で可能となる。マルチメディア文書のみならず、ルールや手続きで表されるあらゆる知識を一括に出版し、流通させることができる。一方、読者は、これらの論文から数式をコピーし、自分の持っているシミュレータやビジュアルライザにかけて、著者とは異なる観点からこの数式を評価することができる。

etc.

## 7. ミームプール: パッドの市場アーキテクチャ

### 7.1 Internetを用いたパッド流通

IntelligentPadのカーネルシステムをクロスプラットフォーム対応にすることにより、異なるハードウェア上で稼働するIntelligentPadシステムの間でパッドを自在に交換し利用することは可能となる。これには、各サイトが同一ライブラリを持ち、パッドの貼り合わせ構造情報と状態情報のみを転送する交換方式から、クラス・マイグレーションを伴う方式までいくつかのレベルが考えられるが、本論文ではパッドのセーブ形式表現のみを転送する前者の方式に話を限定する。パッド・プールの構築には、パッドのクロスプラットフォーム転送機能のみでは不十分である。あらゆるパッドをブラウジングすることができ、パッドの売買や交換が行える場をさらに提供する必要がある。このような場は市場やバザールに相当するものであり、これを







Internetのような世界規模のネットワーク上に構築する必要がある。

著者は、これを4段階に分けて実現しようとしている。第1段階では、MosaicやNetscapeをパッドのカatalogとして用いてパッドの流通を行う。第2段階では、IntelligentPadシステム自身をInternetのフロントエンドとして用いる。第3段階では、これらの方法を統合する。第4段階では、第3段階の成果に、課金、料金請求、支払の機能を組み込む。第1段階と第2段階に関しては、既に技術課題を解決し、システム開発を終了している。第3段階に関しても既に技術課題を解決し、システムを開発中である。

## 7.2 MosaicやNetscapeを用いたパッド流通

図3の左側のウィンドウは、Netscapeをパッドカatalog

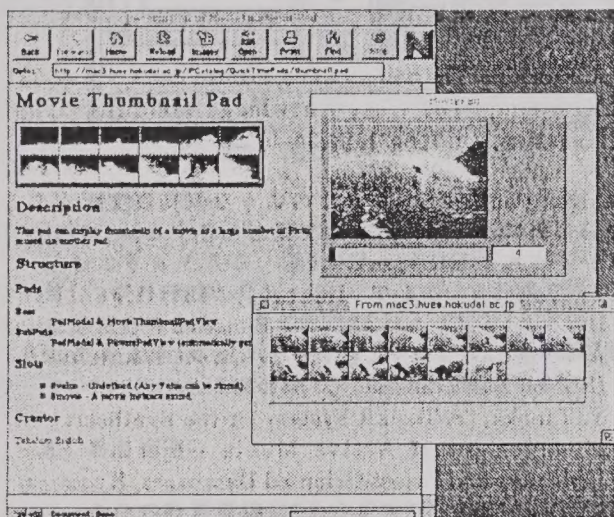


図3. Netscapeをパッド・カatalogとして用いたパッドのネットワーク流通

として用いた例を示している。このカatalog中のMovieThumbnailPadの絵をクリックすると同図右下に示す新しいウィンドウが開かれ、遠隔ファイルから転送されてきたVideoThumbnailPad自体のコピーがこのウィンドウの中に現れる。同図にはこのパッドをユーザサイトが持つビデオに適用した結果が示されている。

Netscapeをパッドのカatalogとして用いるには、対象パッドのディスプレイ・イメージをGIFファイルとして作成しておき、対象パッドの説明文書をHTMLで記述すればよい。この説明文にはGIFファイルも埋めこみ、対象パッドのディスプレイ・イメージもNetscape上に表示できるようにしておく。

Netscapeを用いて、ある遠隔WWWサーバのパッドカatalogをアクセスし、その中のあるパッドの絵をクリックすると、この絵をアンカーとして登録してあった参照リンクがNetscapeシステムに得られる。このリンクは遠隔サーバ中のファイルを指すことができるURL(Universal Resource Locator)で指定されており、Netscapeはこれを用いてこのパッドのセーブ形式ファイルを遠隔アクセスする。Netscapeにはリンク先のファイルとして、任意のファイルを指定することができる機能が用意されており、そのファイルのファイル名の拡張子の違いに応じて、このファイルの遠隔読み出しと共にNetscapeクライアント側のシステムで、このシステムに予め登録しておいた任意のプロセスを立ち上げる機能が用意されている。セーブ形式ファイルの拡張子をPADと決めておき、この拡張子のファイルが参照されたとき、クライアント側でIntelligentPadのパッドローダを立ち上げ、読み出されたファイルをこのローダに渡すようにしておくことができる。ローダはこのセーブ形式のファイルから、対応する合成パッドを再構築する。こうしてパッドがサーバ側のサイトからクライアント側のサイトへと転送される。

## 7.3 IntelligentPadによるInternetのフロントエンド・システム

IntelligentPadでは、いろんな機能をもった部品パッドが用意されている。クリックされると、予め登録しておいたURLを用いて遠隔ファイルをアクセスし、そのファイルにセーブ形式で格納されているパッドをディスプレイ上にポップ・アップするパッドも既に用意されている。このパッドをURLアンカーパッドと呼ぶ。ビデオパッドの上でフレーム毎に位置とサイズを変化することができるオブジェクト・トレース・パッドもすでに用意されている。

例えば、ビデオパッドを用い、ビデオ中の幾つかの対象物に対し、その動きと大きさの変化に応じて常にこれらを覆うように指示した透明なオブジェクト・トレース・パッドを必要な数だけ用意し、これらのオブジェクト・トレース・パッドの上に透明なURLアンカーパッドを貼り、各々が覆っている対象物に関する詳細情報やツールを提供するパッドの格納ファイルを指すURLを忍ばせることができる。図4はこのようにして作った札幌市の観光案内で、空撮されたビデオの中で、旧北海道道庁を覆うオブジェクト・トレース・パッド上に貼られたURLアンカーパッドをクリックすることにより、旧北海道道庁の写真がポップ・アップされたところを示している。

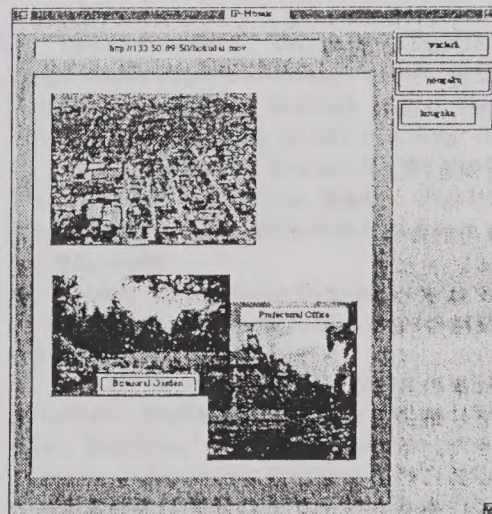


図4. IntelligentPadによるInternetのフロントエンド・システム

このような仕組みにより、Internet上で遠隔地のサーバに格納されているビデオパッドや各種ツールパッドを次々と辿って気に入ったパッドや必要なパッドを手元を持てることが可能になる。このようなシステムの直接的応用としては、ネットワーク・ミュージアムやネットワーク・マルチメディア・ライブラリ、バーチャル・モールなどが考えられる。

## 7.4 HTMLビューワ・パッド

MosaicやNetscapeと互換性のあるHTMLビューワをパッドとして開発することも可能である。図5は、X Window上のあらゆるウィジェットをシステムティックにパッドに変換できるようにしたウィジェット版IntelligentPadと呼ばれる、著者の研究室で開発された新しいバージョンのIntelligentPadを用いて開発されたHTMLビューワ・パッドの使用例を示している。HTML文書の中に、特殊タグを付けて任意の合成パッドを埋め込むことができ、埋め込まれたパッドはHTMLビューワ・パッドの上で、対象テキスト中の埋め込み位置にパッドとして貼られる。テキスト中には貼られるパッドのサイズに一致した空欄が設けられ、この空欄の上にパッドが貼られる。テキストのスクロールに従って、空欄も移動するが、これに対応して、上に貼ったパッドをHTMLビューワ・パッドの上で移動させる。こうすることにより、貼り込まれたパッドがテキストと一緒にスクロールするようになる。







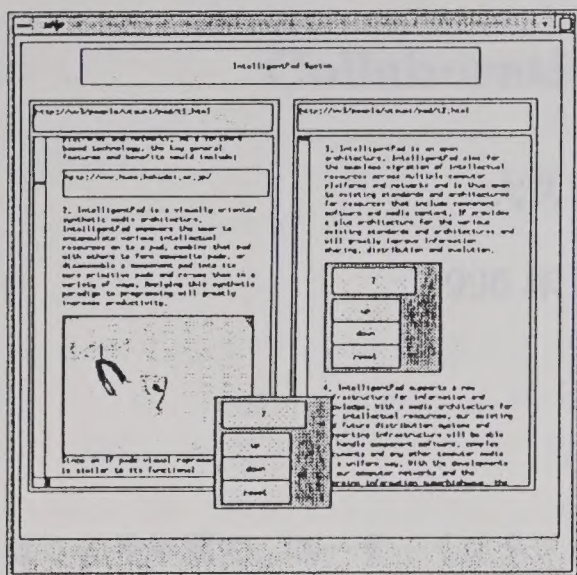


図5. HTMLビュー・パッド

HTMLビュー・パッドはパッドであるから、HTMLビュー・パッドの上にHTMLビュー・パッドを貼り込むこともできる。図5では、HTMLビュー・パッドの上に、2つのHTMLビュー・パッドが2つの欄を構成するように貼られており、右のHTMLビュー・パッドの上にはさらにテキスト・パッドとビデオ・パッドが、左のHTMLビュー・パッドの上には合成パッドが貼られている。

#### 8. あとがき

今日の計算環境は“ネオダマ”に向かって発展しているといわれる。ネはネットワーク、オはオープン・アーキテクチャ、ダはダウンサイジング、マはマルチメディアを指す。異機種種のPCやワークステーションをネットワークで繋いで統合し、マルチメディア対応のGUIを持ったシステムを適用対象のスケールに合わせてライトサイジングでボトムアップに構築できることが望ましいと考えられるようになってきた。そのために、システム構成の変更や拡張に速やかに容易に対応でき、既存のアプリケーションやデータも含めて、ネットワーク上に分散しているデータやアプリケーション等の資源を自在に組み合わせてシステムが構築できるような環境が望まれている。

一方、ソフトウェア開発者の側でも、多様なニーズに速やかに対応できるように、システム機能を標準的なコンポーネントに分け、その組合せで各種のシステムが自在に構築できるような開発環境の実現が望まれている。

このようなことから、ソフトウェアをオブジェクトとしてコンポーネント化し、ネットワークに跨って分散している各種のオブジェクトを統合管理し、それらの間に自在に連携動作を定義できるような分散オブジェクト環境の開発が望まれるようになった。

IntelligentPadはまさにこのような時代の要求にマッチしたシステム・アーキテクチャである。

IntelligentPadにおけるプリミティブパッドの機能の粒度は任意であるが、独立した機能は独立なジェネリックなパッドとして実現するというのが原則である。これにより、再利用性が高まるだけでなく、流通合成パッドの再編集可能性が高まる。

IntelligentPadのアーキテクチャ設計において、著者は汎用性と拡張性を失わない範囲で、基本メカニズムをできるだけ単純にし、いろんな面から制限をできるかぎりきつくすることを目指した。著者は何をしてもよいという完全な自由の中からは文化が育ちにくいと考えている。何をしてもいいということは、何をしていいか、何から始めていいかわからないということにつながっている。できることが少ない範囲に制限されている場合には、あれこれと遊んでいる間に、思いがけない組合せができたりする。それ

が新しい創造へと育つ種となる。我が国では、古来、華道、茶道、書道、武道など、多くの文化が「道」の文化として栄えたが、それらは制限する所から生まれる創造の文化ではなかったであろうか。IntelligentPadもまた、制限する所から生まれる創造の文化をねらっている。

Internetを用いたパッドの流通基盤の構築はミーム・ブールを形成し、模倣と創造を促し、パッドにのった知的資源の進化を加速するであろう。そのためには、部品パッドの種類が何百、何千種類という世界を速やかに実現していかなくてはならない。このような世界の構築にはポジティブ・フィードバックがかかるものである。ポジティブ・フィードバックがかかり始めるようになるクリティカル・マスまで、どのように基盤を整備するかがIntelligentPadプロジェクトがかかえる今後の最大の課題である。

#### 参考文献

- (1) R. Dawkins, The Selfish Gene, Oxford Univ. Press, 1976
- (2) M. Stefik, "The Next Knowledge Medium," The AI Magazine, vol. 7, no. 1, pp.34-46, 1986.

IntelligentPadに関しては以下のような既発表文献がある。これら以外に、応用に関する文献も存在する。

- (3) Y. Tanaka, and T. Imataki, "IntelligentPad: A Hypermedia System allowing Functional Composition of Active Media Objects through Direct Manipulations," IFIP '89, San Francisco, pp.541-546, Aug. 1989.
- (4) Y. Tanaka, "A Toolkit System for the Synthesis and the Management of Active Media Objects," Proc. of Deductive and Object-Oriented Databases, Kyoto, pp. 76-94, Dec., 1989.
- (5) Y. Tanaka, "A Synthetic Dynamic-Media System," Proc. of Int'l Conf. on Multimedia Information Systems, Singapore, pp.299-310, 1991.
- (6) Y. Tanaka, A. Nagasaki, M. Akaishi, and T. Noguchi, "Synthetic Media Architecture for an Object-Oriented Open Platform," Personal Computers and Intelligent Systems, Information Processing 92, Vol III, (ed. F. H. Vogt), North Holland, pp.104-110, Madrid, 1992.
- (7) 赤石美奈, 田中諒, "IntelligentPadにおける演劇機能の実現," 情報処理学会論文誌, vol. 33. no. 6, pp.791-801, 1992.
- (8) 長崎祥, 田中諒, "シセティック・メディア・システム IntelligentPadにおける協調作業場の実現," 情報処理学会論文誌, vol.34. no. 5, pp.1177-1187, 1993.
- (9) 田中諒, "シンセティック・メディア・アーキテクチャとその展望," 電気学会誌, vol. 113, no. 6, pp. 101-108, 1993.
- (10) 赤石美奈, 田中諒, "IntelligentPadにおける部分構造検索," 情報処理学会論文誌, vol.35, no.2, pp.232-242, 1994
- (11) 長崎祥, 田中諒, "シンセティック・メディア・システム: IntelligentPad," コンピュータソフトウェア, vol.11. no.1, pp.36-48, 1994.
- (12) 赤石美奈, 田中諒, "部品のあいまいな位置に関する検索," 情報処理学会論文誌, vol.36, no.8, pp.1916-1925, 1994.







# Collaborative ZipEditor

大久保清貴

1996 年 3 月 19 日

## 1 概要

ZipEditor はこの半年間で、ユーザ・インタフェースはあまり変化していないが、中身は大きく変わった。

まず Zip のファイルを管理する部分をクライアント部分とサーバ部分に切り離した。その結果 ZipEditor は3つのコンポーネントから構成されることになり、これらはそれぞれ独立したサイトで動作させることが可能となった。それぞれのコンポーネントがインタフェースをとるユーザは図1の通りである。

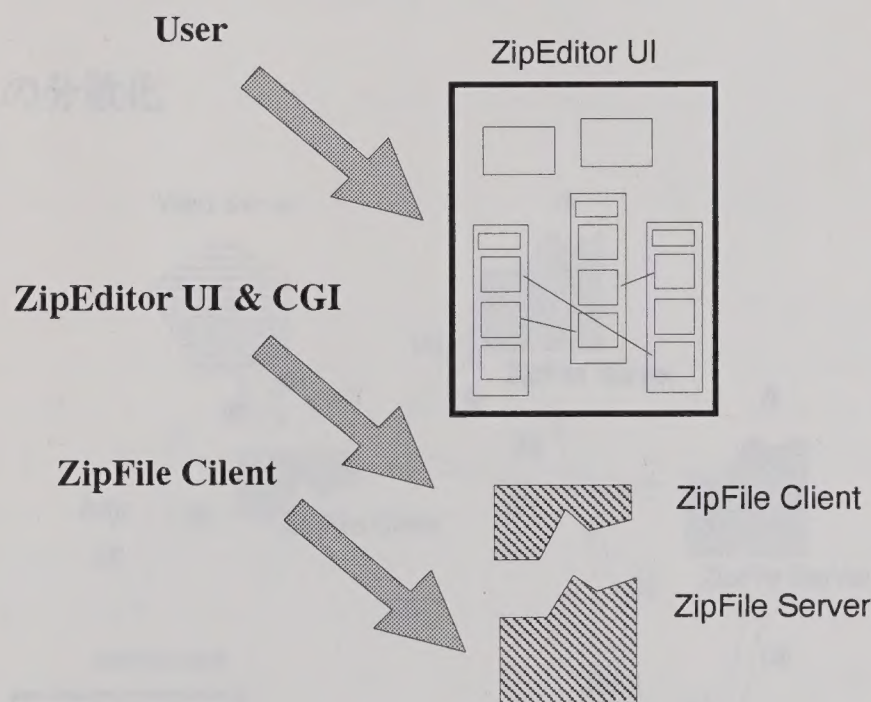


図 1: ZipEditor のコンポーネント

コンポーネントが独立すると、ZipEditor が扱うデータ自体もサイトをまたがってアドレスできるようにする必要がある。ZipEditor は、zip と item の識別子を URL(Universal Resource Locator) のようにホストのドメイン名とポート番号を頭に付けることで対処した。

ZipFileManager は Tcl(Tool Command Language) を使ってプログラムしているので、Tcl インタープリタに TCP/IP を用いて通信を行うためのコマンドを追加した。従って、コンポーネント間のプロトコルは Tcl コマンドからなる文字列である。

上に述べた分散化とそれに伴う共同協調作業環境への拡張は今回の目玉であるが、もう一つの柱は WWW(World Wide Web) とのコネクションである。







WWW の急速な広まりに便乗することを目指し、また Web の世界に欠けているドキュメント・パーツのフェアな再利用が transclusion で可能になることを知ってもらうために、ZipEditor で作ったドキュメントを HTML ページにダイナミックに変換するツールを作った。これにより ZipEditor 自体をダウンロードしないでも、zip のデータ構造を Web ブラウザを通して体験できるようになった。ZipEditor で作ったものは、通常の Web ページと異なり、各パラグラフを再利用している別のドキュメントへ飛んだり、どのドキュメントをベースにそのドキュメントを作ったか、あるいは各パーツが誰によって書かれたか等を、アンカーをクリックするだけで調べることができる。

Web ブラウザから ZipFile サーバの管理するデータを見る機能は、ZipEditor 自体にとってもそれがまだ持っていない機能、たとえば zip や item の著者、作成日、どのサーバ上にデータがあるか等を調べる機能を提供している。そこで Web との関係を強めるために、Netscape ブラウザをリモート・コントロールする機能 (CCI の openURL) を使って、ZipEditor に表示された zip や item から turn browser page というメニューを選ぶと Netscape ブラウザがその zip や item のページを開くようにした。

ZipEditor は今回の進歩により、インターネット上のどこからでもアクセス可能になり、WWW に近いスケール拡大性を持つようになった。しかし他者が提供する Web ページが読みだし専用であるのに対し、ZipEditor は書き込みが可能である。これにより共著などが可能になり利用範囲が拡大したが、同時に新たな問題も生じた。

## 2 ZipEditor の分散化

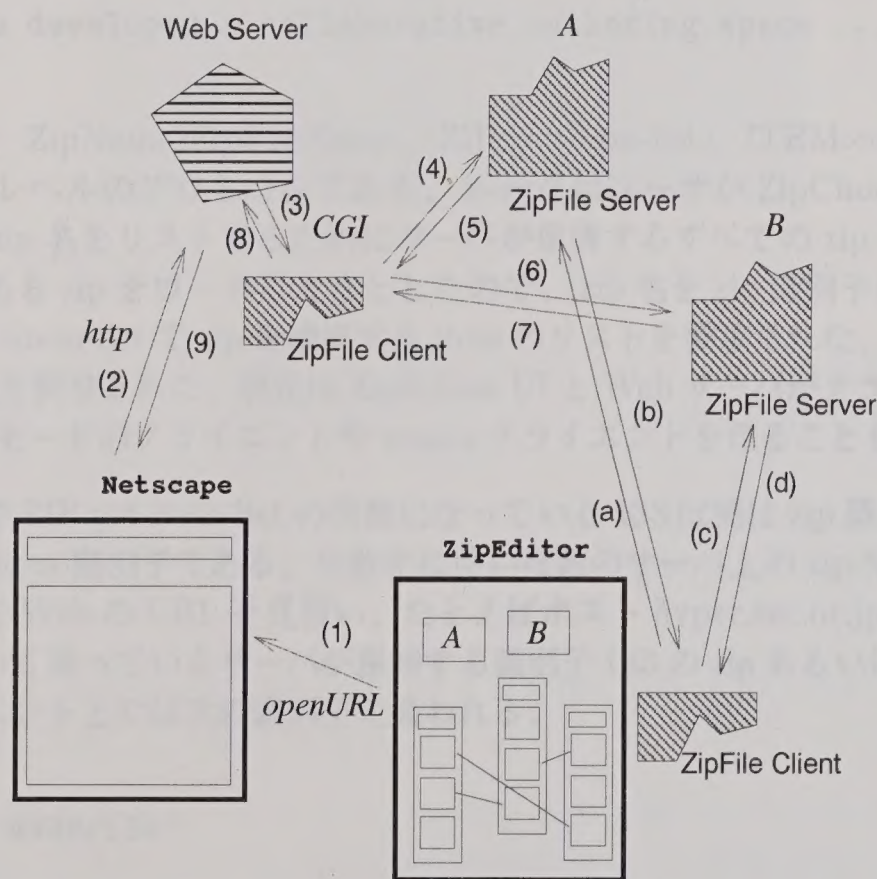


図 2: 分散化した ZipEditor のアーキテクチャ

分散化した ZipEditor のアーキテクチャを図 2 に示す。ZipEditor の UI は C++ を使って IntelligentPad EvaSys0.95 上にインプリメントしている。ZipFile Manager はクライアント側とサー







バ側に分離され共に Tcl で書いている。クライアント側は ZipEditor UI から呼ばれるときは ZipEditor UI と結合した 1 つのプロセスである。Web サーバの CGI(Common Gateway Interface) から実行されるときは ZipFile クライアントだけで 1 つのプロセスとなっている。このプロセスは Web サーバと同じサイトで実行される。

ZipFile サーバは個々に独立したプロセスで、ZipEditor UI や Web サーバのサイトとは異なるサイトで実行できる。ZipFile サーバは ZipEditor UI から実行された ZipFile クライアントから呼ばれたのか、CGI によって起動された ZipFile クライアントから呼ばれたのか区別しない。クライアント／サーバ間のプロトコルは Tcl コマンドである。サーバに要求されたメッセージとその返答メッセージの一部を示す。

```
Starting on mindmachine4.fantastica.com IP addr = 192.251.56.4 Port = 9876
Connection accepted from 192.251.56.4 1148
5>> ls-zip2
5<< ZipEditor.new ZipEditor biocomputer.txt temporary unlabeled0015.0
128 unlabeled0015.0042 unlabeled0039
5>> ZipName2ZipFileName ZipEditor
5<< 133
5>> ZIP:get-item-list 133
5<< 134 135 136 137 138 139 140 141 142 143 144 145
5>> ITEM:content 134
5<< ZipEditor: A Collaborative Workspace for Sharable ....
5>> ITEM:content 135
5<<      We have developed a collaborative authoring space ....
```

ここでは、ls-zip2、ZipName2ZipFileName、ZIP:get-item-list、ITEM:content が HTTP の GET、HEAD などと同じレベルのプロトコルである。ls-zip2 はユーザが ZipChooser の Rebuild ボタンを押したときに zip 名をリストするためにサーバが保持するすべての zip 名を返すよう要求した。次にユーザがある zip をロードしようとしたので、zip 名を zip 識別子に変換せよと要求された。次に ZIP:get-item-list で zip を構成する item のリストを要求された。その後 ITEM:content で item 内のデータを要求された。現在は ZipEditor UI と Web サーバがクライアントであるが、WWW と同様に行モードのクライアントや emacs クライアントを作ることでもある。

上のトレース出力で ZIP:get-item-list の引数になっている 133 は実は zip 識別子である。そのあとの 134 や 135 は item 識別子である。分散化に伴い複数のサーバ上の zip や item を識別する必要がある。そこで Web の URL を見習い、たとえばホスト hyper.sec.or.jp のポート 9999 番にコネクションを開いて待っているサーバが保持する識別子 133 の zip あるいは item は、他のサーバあるいはクライアント上では次の識別子で扱われる。

```
hyper.sec.or.jp:9999/133
```

今回、分散化に対応して ZipEditor ユーザ・インタフェースから見た変更点は ZipChooser パッドの上の方にサーバのドメイン名とポート番号を入力するためのパッドが追加されたのと、複数の ZipChooser が開けるようになった点と、Netscape ブラウザをリモート・コントロールするメニューが追加されたことの 3 つだけである。







図 3では ZipChooser が 2つ開いてる。ZipEditor を起動したときは、ユーザがデフォルトとして設定した環境変数の値を ZipFile サーバ名とポート番号とする ZipChooser が現れる。ユーザは Zap パッドのメニュー new zipchooser を選んで複数の ZipChooser を呼び出せる。これらが保持する zip 名のリストからこれまで通り選んで、zip をロードできる。ユーザはどのサーバに zip や item があるか意識せずに操作できる。

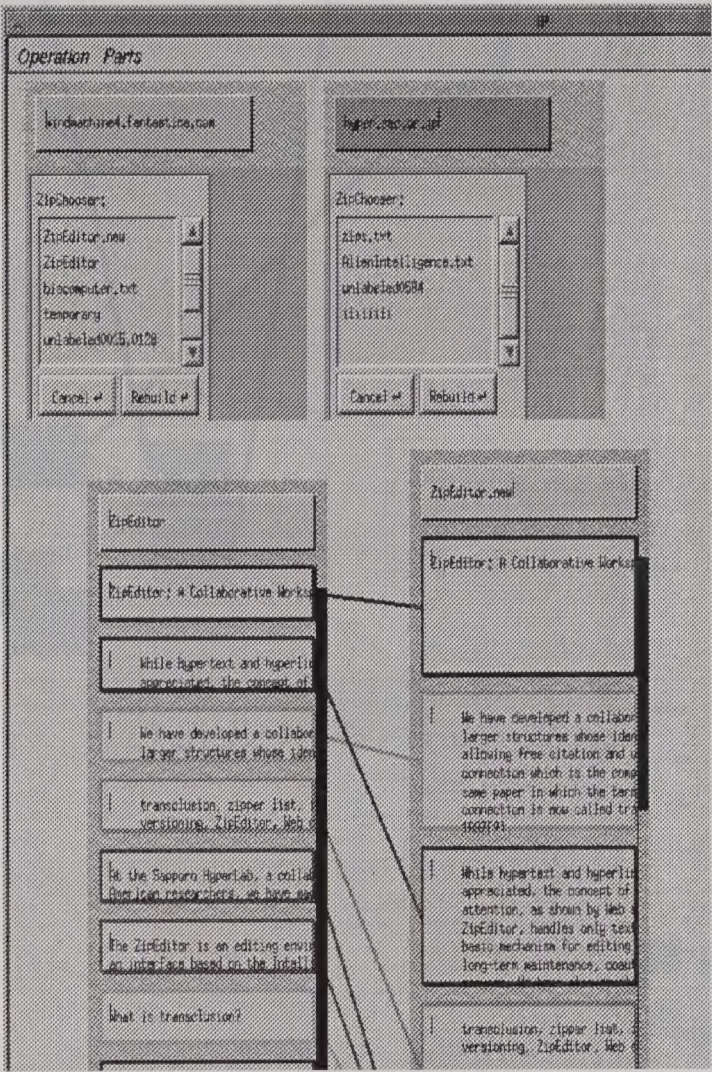


図 3: 複数の ZipChooser によって複数のサーバを扱う

たとえばある zip や item から派生したドキュメントを作るときは、これまで通り copy をして始める。そうすると派生したドキュメントは親と同じサーバに保存される。ただし、新規の zip または item を作るときは親がないのでサーバを選択しなくてはならない。ZipChooser の Rebuild ボタンを押すとサーバ名をもった欄が金色になり、これが新規の zip と item が格納されるサーバであることを示す。図 3は白黒なので分かりにくいですが、左側の色の薄い方のサーバが選択されている。

上で派生させて作った zip や item は親と同じサイトに作られると言ったが、これがもっとも簡単な方法だったからである。この様子を図 4に示す。

しかしこの方法には欠点がある。Web サーバと同じように個人が ZipFile サーバを立ちあげるとすると、自分が書いたドキュメントの断片を保持するサーバの管理者がサービスをやるかもしれないからである。Web サーバが頻繁にサービスを変更したり、他のサイトやネットワークに移動している現状を考えるとこの方法はまずいと思う。

ただし、現在の方式にも 1つの大きなメリットがある。派生した zip や item は必ず同じサーバで管理されるので、派生関係を知るためにインターネット上のすべてのサーバをスキャンする必







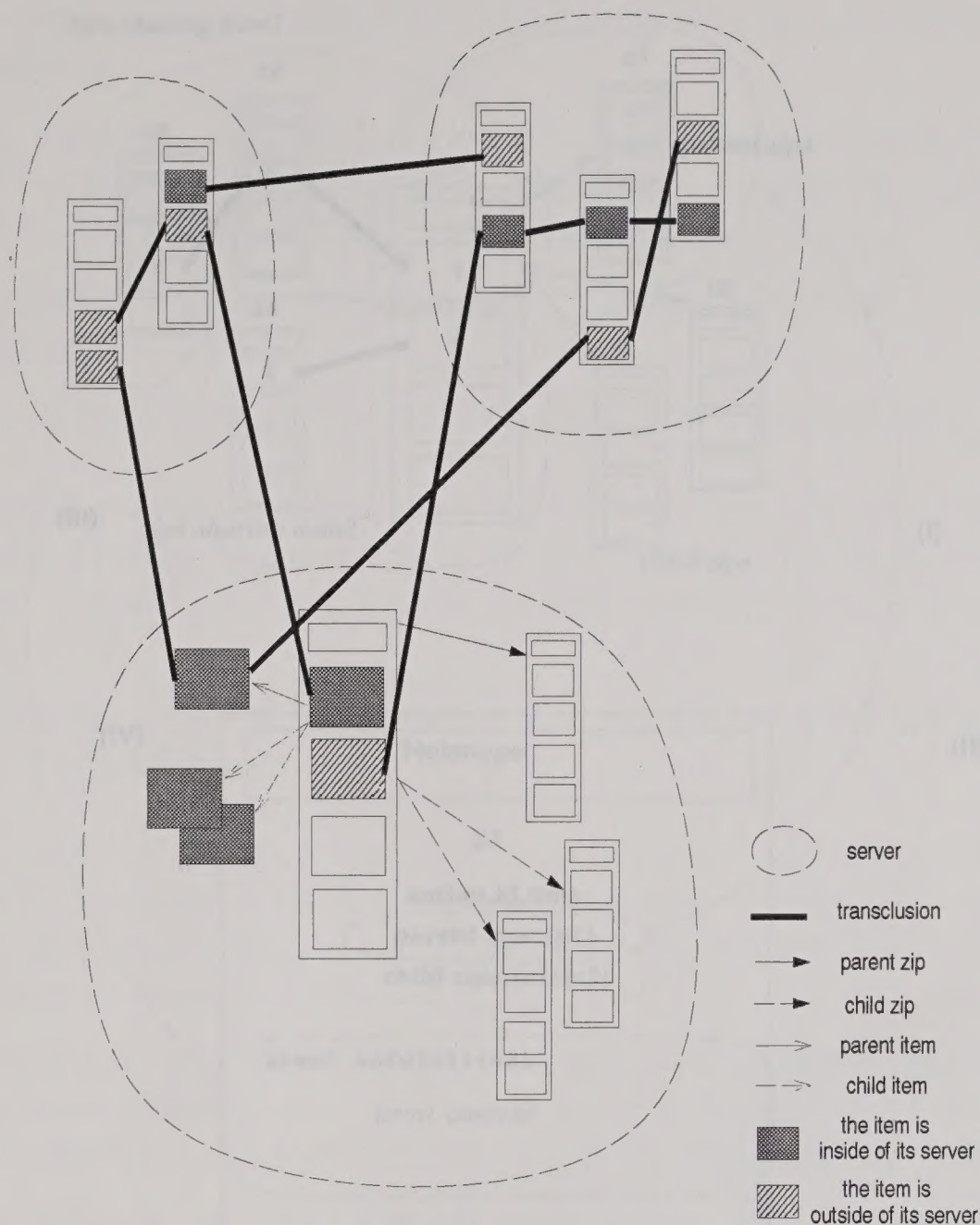


図 4: zip と item がどのサーバに作られるか？

要がない点である。トランスクルージョンの関係は集中管理されていないので、ある item を再利用しているすべて zip を求めようとすると、現在 Web サーチ・エンジンが行っているようなスキャンをする必要がある。

一応現在でも、あるサーバがダウンしても、そのサーバが管理している item と zip が扱えないだけで、サーバが復帰した後 ZipChooser の Rebuild ボタンを押せば再び使えるようにはなっている。

### 3 ZipFile の Web 上へのパブリケーション

図 2 に示したように ZipEditor UI から zip をアクセス場合に比べて、Netscape ブラウザを通してアクセスする場合はかなり複雑である。この図は、たとえばサーバ A 上に zip があってその一部の item がサーバ B にある場合の通信の流れを表している。

ZipEditor UI を使って、zip 上で turn browser page を選択したときは次のことが行われる。







zips sharing item1

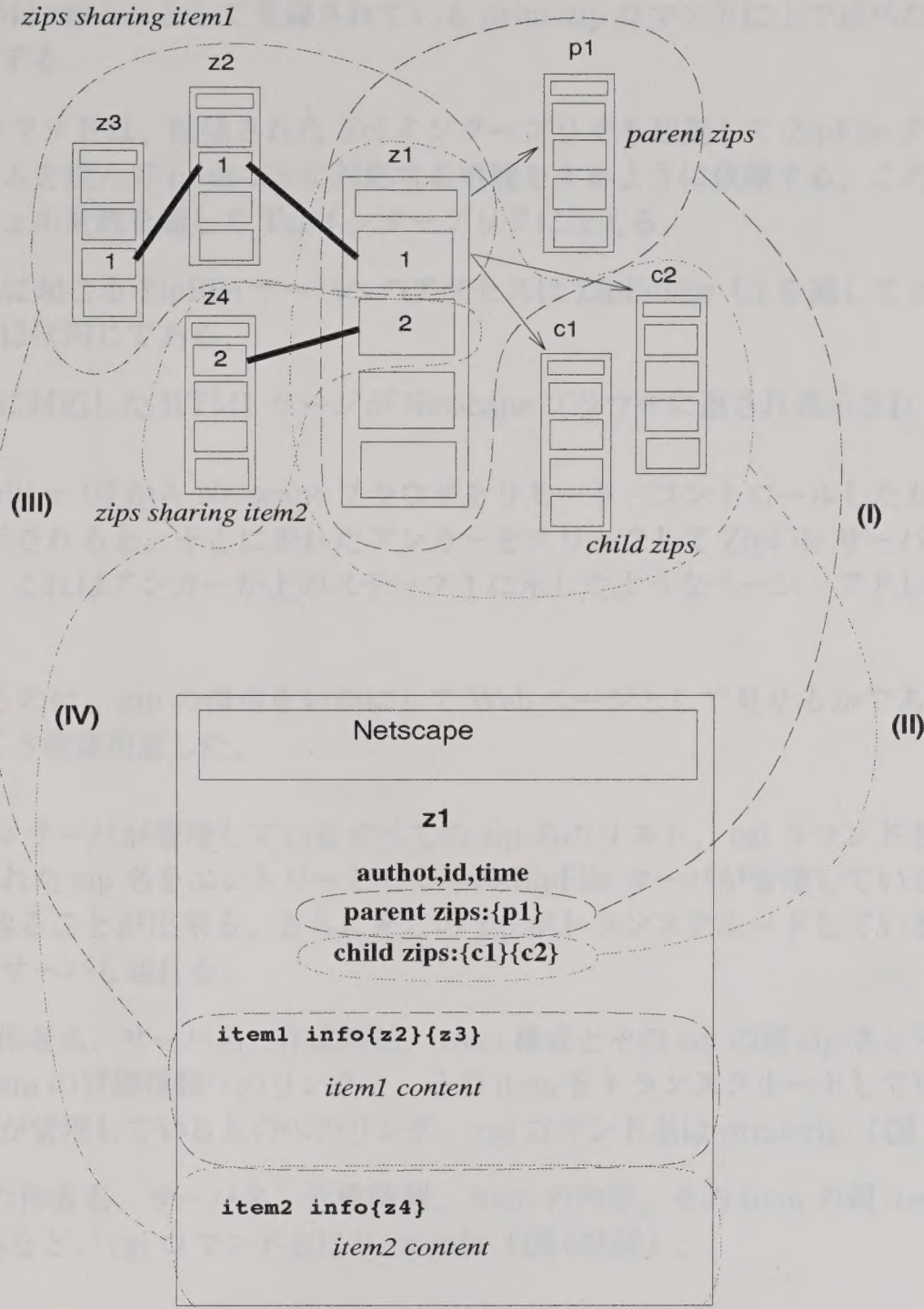


図 5: Zip の HTML ページへのマッピング

1. X ウィンドウ・サーバの管理情報から現在立ち上がっている Netscape のウィンドウ識別子を求め、その Netscape ブラウザに対してたとえば次のようなページを開くよう openURL を送る。

```
http://mindmachine4.fantastica.com:9090/cgi-bin/print-zip?ZipEditor+mindmachine4.fantastica.com/133+0
```

このページ・アドレスの意味は mindmachine4.fantastica.com という名前のホストでソケット 9090 番を聞いている Web サーバに対して、print-zip という cgi コマンドにパラメータ ZipEditor と mindmachin4.fantastica.com/133 と 0 を渡してダイナミックに HTML ページを作れと言っている。2 番目の引数は zip 識別子である。

2. Netscape ブラウザは、Web サーバ mindmachine4.fantastica.com:9090 に上のページ・アドレスを伝える。







3. Web サーバは cgi-bin として登録されている print-zip コマンドに上で述べた3つの引数を渡して起動する。
4. print-zip コマンドは、拡張された Tcl インタプリタを起動して ZipFile クライアント用のプログラムを読んで print-zip に対応する処理をするように依頼する。このとき上で述べた引数はシェル変数を通して Tcl インタプリタに伝える。
5. ここから先に起こる ZipFile サーバへのアクセスは ZipEditor UI を通して zip をアクセスする場合とほぼ同じである。
6. やがて zip に対応した HTML ページが Netscape ブラウザに返され表示される。

いまの場合 ZipEditor UI から Netscape ブラウザをリモート・コントロールしたが、一旦 zip がブラウザ上に表示されると、そこに表れたアンカーをクリックして ZipFile サーバをアクセスできるようになる。これはアンカーが上のステップ1に示したようなページ・アドレスを持っているからである。

ここで問題になるのは、zip の構造をいかにして Web ページとして見せるかである。今回ページのタイプとして3種類用意した。

- ある ZipFile サーバが管理しているすべての zip 名のリスト。cgi コマンド名は ls-zip。これで表示された zip 名をエントリーとしてその ZipFile サーバが管理しているすべての zip と item を辿ることが出来る。さらにその zip がトランスクルードしている item を管理している別サーバも辿れる。
- ある zip の作者名、サーバ名、作成時刻、item 構成とその zip の親 zip 名と子 zip 名の表示。さらに各 item の詳細情報へのリンクと、その item をトランスクルードしている zip の内、そのサーバが管理しているものへのリンク。cgi コマンド名は print-zip（図5参照）。
- ある item の作者名、サーバ名、作成時刻、item の内容、その item の親 item の内容、子 item の内容など。cgi コマンド名は item-info（図6参照）。

## 4 問題点と課題

ここで述べたシステムはまだ実験として使ってもらうレベルである。実用的に使うには次の問題をクリアしなければならない。

### ● パフォーマンス

パフォーマンスの問題としてまず重大なのが、1文字入力毎に全画面を書き直している点である。IntelligentPad EvaSys の大きな欠陥である。ZipEditor UI については開発時間を短縮するために C++ を使わず Tcl ですました部分がある。ZipFile サーバについていうとシングル・スレッドであるので Web サーバのように複数のコネクションに対して同時にサービスできないということがある。サーバは大部分 Tcl で書いてあるのでマルチ・スレッド化が今のままでは難しい。Tcl 自体も大変遅い言語である。また Web ページとしてパブリッシュする処理で、item をトランスクルードしている zip を求めるのにサーバ内の zip を全スキャンしている。ここも item や zip の親子関係についてやったようにインデックシングすべきである。







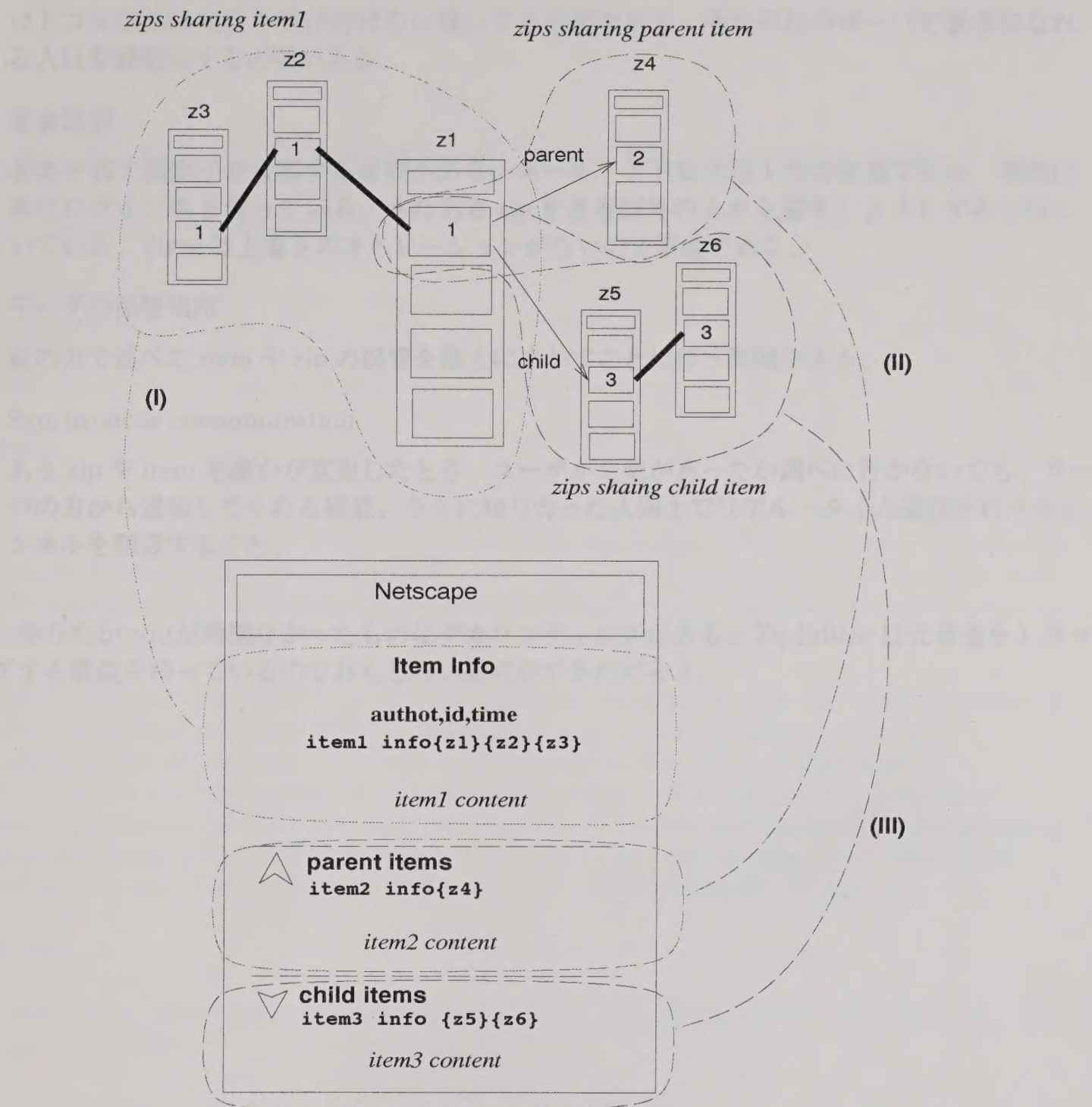


図 6: Item の HTML ページへのマッピング

- 信頼性

ZipEditor UI についてはかなりクラッシュが減ったがまだまだ落ちるケースがある。ZipFile サーバについては正常なデータについては信頼性が高いが、誤ったデータを渡されると弱い。また Web サーバのように stateless でないので、リカバリーが難しいということもある。

- security/authentication

この問題については取り組みたかったが時間がなかった。現在 ZipFile サーバは誰がコネクションを張ったかチェックしていない。さらに ZipFile サーバは通常の Tcl コマンドをすべて実行するのでこれを通して login することもできる。したがって ZipFile サーバを通常のユーザ・アカウントで運用するのは危険である。コネクションの相手を同一ネット内に限るような仕組みを入れた方がよいだろう。また ZipFile クライアント／サーバ間のプ







ロトコル以外は実行を受け付けない様にする必要がある。それぞれのサーバで著者になれる人は登録制にする必要がある。

- 著者識別

著者を表す識別子を考案する必要がある。メール・アドレスは1つの候補である。現在は単にログイン名を使っている。またある zip を著者以外の人が上書きしようとするとはじている。Item は上書きのオペレーションがないので安全である。

- データの保管場所

前の方で述べた item や zip の保管を他人にまかすことに伴う問題がある。

- Synchronous communication

ある zip や item を誰かが変更したとき、ユーザが変更があったか調べに行かないでも、サーバの方から通知してくれる機能。さらに知り合った人同士でリアル・タイム通信を行うチャンネルを開設すること。

今回、やりたかったが時間なかったものにアカウンティングがある。ZipEditor は元著者をトラッキングする機能を持っているのでおもしろい研究ができただろう。

## Keywords

translation, ZipEditor, hypertext, transclusion, World Wide Web, Zip5, Zip6, Web connection

## Introduction

In the Sapporo HyperLab, a collaboration research effort among Japanese and American researchers, we have made the development of a new form of hypermedia one of our principal goals.

The ZipEditor is an ongoing development which carries out transclusion, employing an intelligent browser. In this paper we will introduce transclusion, explain how the ZipEditor implements transclusion, and briefly discuss its contribution to the World Wide Web.

## What is transclusion?

A hypertext, as most textbooks or references of people, is a document which may be divided into an arbitrary number of pages or sections. In this paper, we will introduce transclusion, explain how the ZipEditor implements transclusion, and briefly discuss its contribution to the World Wide Web.







# ZipEditor: A Collaborative Workspace for Sharable Material

Kiyoki Ookubo; HyperLab; Sapporo, Japan.

ookubo@hyper.sec.or.jp

Theodor Holm Nelson; HyperLab; Sapporo, Japan.

ted@hyper.sec.or.jp, ted@xanadu.net

## Abstract:

We have developed a collaborative authoring space for text pieces and larger structures whose identities are permanently maintained, while allowing free citation and update. The basic idea behind this, a form of connection which is the complement of hyperlinks, was introduced in the same paper in which the term "hypertext" first appeared[1]. This form of connection is now called *transclusion* (a word which first appeared about 1987[2]).

While hypertext and hyperlinks are widely accepted, understood, and appreciated, the concept of transclusion has received only limited attention, as shown by Web searches [6], [7].

Although our system, named *ZipEditor*, handles only text, it shows the strength of transclusion as a basic mechanism for editing documents which require intercomparison, long-term maintenance, coauthoring, and distribution of their parts among servers. We have also developed a World Wide Web interface to give users a preliminary experience of the kinds of parallel-coupled documents that may be created with such an editor.

## Keywords:

*transclusion, zipper list, implied connection, side-by-side visualization, versioning, ZipEditor, Web connection*

## Introduction

At the Sapporo HyperLab, a collaborative research effort among Japanese and American researchers, we have made the development of a distributed zipper-list[2] editor one of our principal goals.

The ZipEditor is an editing environment which supports transclusion, employing an interface based on the IntelligentPad System [8]. In this paper we will introduce transclusion, explain how the ZipEditor implements transclusive structure, and finally discuss its ramification to the World Wide Web.

## What is transclusion?

A hyperlink, as now familiar to millions of people, is a connection which may be followed from an anchor or span to some other portion of data, and perhaps followed in the reverse direction as well. In the case of the World Wide Web, such a starting anchor is typically an underlined word or part of a graphic image, typically leading to a related HTML page; there is no support for navigation in the opposite direction. However, other hypermedia designs, such as the Xanadu(R) system [2], attach the ends of links to spans or sections of data, and are designed to allow navigation in both







directions.

Transclusion is a simple concept, but difficult to explain because it represents an unusual way of thinking about information. There are many ways to look at the transclusion relation, but each is simply a pathway to understanding the same concept.

Transclusion is a form of equality, but with special visualization properties which distinguish it from ordinary copies (which we may call detached byte copies), ordinary instances, and virtual copies. None of these are conveniently viewable together for intercomparison.

Transclusion is the complement of linkage. While links connect materials which are different, transclusion connects materials which are **the same**. Transclusion means re-use and inclusion of the same materials in different places, keeping track of the materials' being the same, and showing that they are the same.

A sequential document can be viewed as sequenced by an organizational thread which pierces, and ties together, a series of document pieces. **\*\***[footnote: This may also be thought of as a multi-hop linking method. The World Wide Web does not provide a way to express such a multi-hop thread, but only provides a single-hop linking method. However, such larger sequences can be provided manually by the author in various ways, usually by preparing an index page to represent a natural sequence to read.]

But the same pieces can be threaded in more than one way at the same time, as shown in Figure 1, where item C is on both threads.

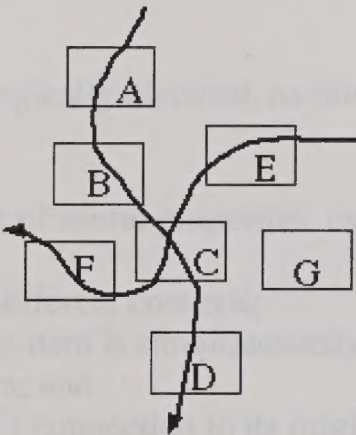


**Figure 1: Document pieces connected into different sequences by separate threads**

In this example there are two threads, with piece C shared between them. Piece C participates in both sequences.

This picture can be rearranged to yield Figure 2, which is logically equivalent to Figure 1.







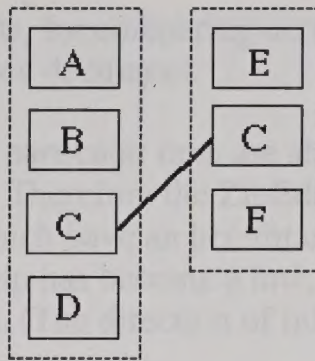


**Figure 2: Document threads represented as side-by-side columns.**

In Figure 2, the two threads of Figure 1 have been replaced by two vertical sequences of items, item C being shared by both sequences. The junction of the two threads, shown in Figure 1, is shown in Figure 2 as a thick line between the two items marked "C."

This figure shows exactly the same structure as Figure 1, differently represented. What the two equivalent pictures have in common is the transclusion relation. This means that the same item, C, exists in both contexts; that the two instances of C are logically *the same* item in both contexts, and that the user knows it. Neither instance of C is a detached byte copy unknown to the other.

We may thus think of transclusion as either:

- the same item in more than one place (an abstract conceptual view), or
- separate instances of a master item viewable in more than one context (an implementation-oriented view).

We deem these two definitions to be logically identical, as illustrated respectively in Figure 1 and Figure 2,

The transclusion relation has a number of useful properties, especially:

- the item can be compared in its different contexts;
- a link made to any instance of the item is simultaneously and equivalently made to all instances and contexts of the item; and
- since this relation tracks an item's connection to its original author, it can be utilized to solve the copyright problem mentioned in [10] democratically [3][4]. This is important in an open network environment like the Internet.

The transclusion connection is not created manually like a link, where the user's explicit action joins two items. Instead, transclusion is an *implied* connection, implicit in the data structure itself.

## The HyperLab ZipEditor

The HyperLab ZipEditor is a collaborative resource for editing and organizing complex parallel structures, shown as divided into zipper lists. Its fundamental design is exactly that proposed by Nelson in 1965 [1]. We have extended it for multiple users and multiple servers. We have also provided an interface for reading and browsing the resulting materials through the World Wide Web.

Our ZipEditor adopts the side-by-side visualization style of Figure 2. This is convenient for







comparing documents which share parts, for comparing derivative and related works, and for comparing different versions of the same document.

However, when different versions of a particular item are shown simultaneously, they are no longer the same, as shown in Figures 1 and 2. Therefore the ZipEditor uses a thinner line to visualize the connection among document pieces which have an hereditary relationship (where one piece is derived from the other). The relationship has become a *link*, rather than a transclusion, since it now connects two items which are different. (The direction of inheritance is not shown in our current version.)

The data objects which the ZipEditor handles are immutable, i.e. they are not overwritten or thrown away. When a data object is updated, the system simply creates a new version of it and makes a note of the hereditary relationship. This nondestructive approach is the same one taken by the various Xanadu system designs (citation). It allows us to avoid various problems, such as obsolete links, lost origins, and the necessity for a complicated concurrency control mechanism, as explained in [9].

The ZipEditor exactly follows the visualization style shown in Figure 2, as developed in Figure 3, which is a screen snapshot. The ZipEditor uses the term *zip* instead of *thread* and *item* for text piece. \*\*[footnote: In the original 1965 paper [1] they were called "zippered lists," which in everyday usage by Project Xanadu and its alumni over the decades has been shortened to "zipper list" and, by 1981, to "zips." A zip-based design for office software was prepared at Datapoint Corporation in 1981, but never implemented [5].]

## User Interfaces

Because the HyperLab has emphasized IntelligentPad and its uses, we have been fortunate enough to be able to use an early prototype of the Unix version of IntelligentPad to implement the interface.

IntelligentPad is a modular component software system very consistently built on a "paste and peel" metaphor. All objects are pads, that is, 2 1/2D combinations of two-dimensional objects pasted one on top of the other, resulting in a functionality which combines the lower pads as subroutines. This top layer may in turn be replicated as a pastable unit, creating a new instance of the conglomerated object.

Mapping the ZipEditor operations to IntelligentPad was a challenge, since the two systems had been designed in absolute independence. But it worked out. The paragraph manipulation functions demanded by the ZipEditor mapped precisely to movable pads, as did the zipper lists themselves. Therefore all the components visible in the author's interface are implemented as pads. Zips are pads, as are the items on them.

Because IntelligentPad allows component pads to be independently manipulated, these visual parts are easy to rearrange.







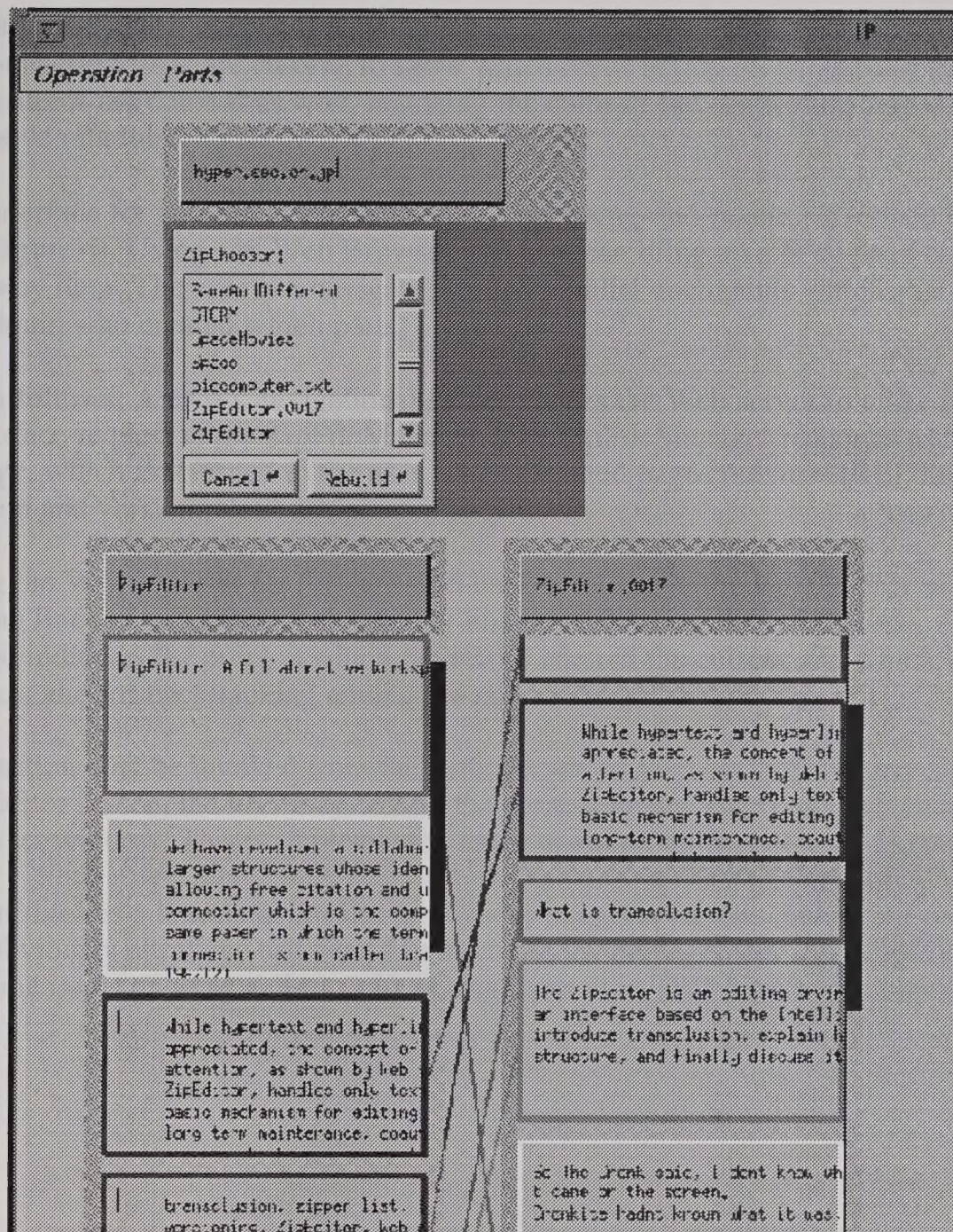


**Figure 3: ZipEditor as Seen by an Author**

The actual author's interface is shown in Figure 3. The pad positioned in the upper left corner of Figure 3 we call the ZipChooser. It shows the address of the server, followed by a list of all zips available on that server. (To facilitate collaborative use, all the zips on a given logical server are merged.)

Below the ZipChooser, in Figure 3, are two particular zips. They have been brought to the screen by choosing them from the ZipChooser.

As many ZipChoosers may appear in the IntelligentPad window as desired, each representing the contents of a logical ZipEditor server. Thus the user can access zips coming from several different servers at once, freely rearranging and revising items and zips from different sources almost without noticing (except for network delay etc.).

Various visual cues are part of the interface. Some pads are translucent, allowing many zips to be placed on the screen while showing as much information as possible. We use color to help users







recognize connected items, even when the screen becomes tangled with many connections between items. The connection-lines are colored to show which items match. In the case of transclusions, the borders of the items are colored as well, matching the color of the connection-line which shows their transclusive relation.

Besides the interface for authors, we have created a browsing-and-reading interface for the World Wide Web (Figure 4). Using any Web Browser, the user can bring up a Web page corresponding to any zip or to any item. The user can navigate through parallel documents which share parts of the text, and ascertain who created which parts.

This works as follows: A user's navigational choice causes the Web server to make a Common Gateway Interface request to the ZipFileManager. The ZipFileManager responds to it by generating an HTML page, which is returned to the user. The generated page contains HREF links matching the transclusion and version relationships.

Besides using the Web browser for forward-and-back scrolling in the zip, the user may jump sideways along lines of transclusion or heredity linkage, following the HREF links. (The generated Web page also indicates the point at which the reader entered the current zip, which is useful for both orientation and for backtracking among the zips of the parallel document.)

This same function may be invoked from the author's screen (Figure 3): From any zip on the author's ZipEditor screen, the user may call up the corresponding Web page, which appears in a separate browser window. This is implemented using openURL of the Netscape Browser.

Using the ZipEditor as a remote controller in this way suggests various interesting possibilities. For instance, an obviously attractive use would be Web hot list management from ZipEditor, easily putting the same items on several lists and tracking these uses. (We have not implemented this, however.)

Figure 4: Zip File as Seen from the Web

## The Implementation

The overall architecture of the ZipEditor is shown in Figure 5. The interface between the client and the ZipFileServer is a remote call which sends a command string (in the Tcl language) to the ZipFileServer.

The ZipFileServer includes

- a server
- a client
- a manager
- a database (via a server)
- a local

Each of these generates a corresponding Tcl command, which is sent to the ZipFileServer. The server updates the database accordingly. Thus the local protocol between the client and the ZipFileServer is implemented by the Tcl interpreter of the local site, as is the body of each of the ZipFileServer















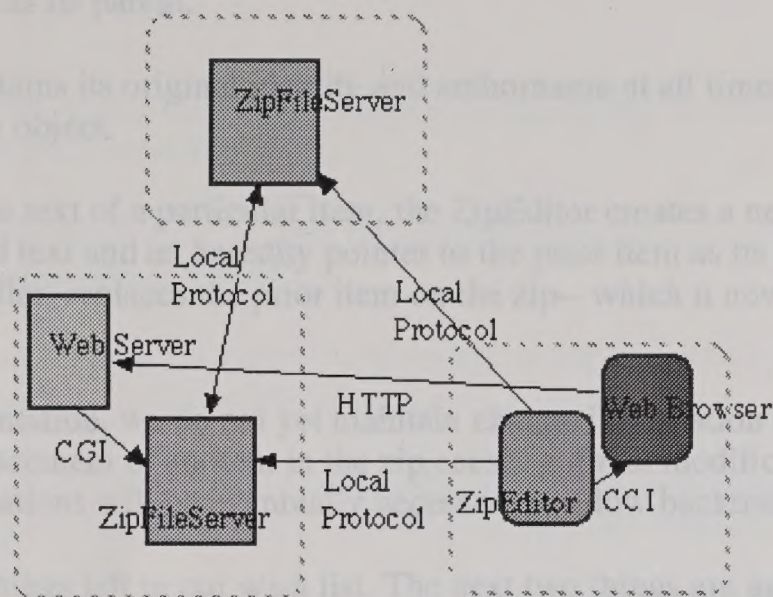


**Figure 5: The Architecture of the ZipEditor**

The ZipFileServer manages a database of items and zips, maintaining their frozen identities and hereditary version relationships.

A zip is represented as a sequence of pointers to items. When the user calls a zip to the screen, the text contained is retrieved item by item. The sharing of an item between two zips --transclusion-- is represented merely as the fact that two zips point to the same item. This is recognized by the server and presented accordingly, with a transclusive connection-line between the item as shown in each of the zips.

The information fields which an item holds are:

- the author name.
- the creation time.
- the parent item identifiers (cross-server parentage is allowed, where an item's origin is listed as being from another server).
- the identifier of the zip on which it was born.
- a text fragment.
- a URL-like location on the Internet, allowing the item to be referenced and manipulated by multiple clients, and by servers other than the one it belongs to.

The information fields which a zip holds are:

- the author name.
- the creation time.
- the parent zip identifiers (cross-server zips are allowed, containing items from more than one source).
- a sequence of item identifiers.
- a URL-like location on the Internet, allowing the zip to be referenced and manipulated by multiple clients, and by servers other than the one it belongs to.

For example, when the user wants to re-use some zip created by another person, he or she makes a copy of the zip using the ZipEditor client. This operation creates a new zip object containing the same sequence of item identifiers as the original zip, with the user's name as the new author, and







identifying the original as its parent.

Each item, however, retains its original identity and authorname at all times. Modification of an item creates a new item object.

As the user modifies the text of a particular item, the ZipEditor creates a new item object, containing the modified text and an heredity pointer to the prior item as its parent. This new item, which has a new identifier, replaces the prior item on the zip-- which it now recognizes as its parent.

In our current implementation, we do not yet maintain change information for reverting to the previous state: the replacement of an item in the zip causes a direct modification of the zip object. Recording change operations will be eventually necessary to allow backtracking.

There are many other things left in our wish list. The next two things are agenda perhaps:

- a simple accounting mechanism for making possible a payment to the original author by each recipient, taking advantage of this aspect of transclusion.
- an introduction of digital signature techniques to prevent tampering, while guaranteeing the free flow of information

## Conclusion

The success of the World Wide Web has increased the information available to us dramatically; and the Web makes our net life more exciting. We think that some form of the mechanism presented here will be an important next step in this healthy progress.

Although people want to re-use others' materials as in-line text, everyone knows the problems this can cause. Transclusion offers a means for enabling such re-use on a cooperative basis, with original ownership retained by the original rightsholder. Transclusion may be thus a fundamental concept for sharing information fairly.

There is another reason to motivate the adoption of transclusion: The future use of the Web for the representation of knowledge. We expect this effort to change the Web scape to a more structured form. Transclusion may be thus the first citizen among the implied connections needed for knowledge representation. Implied equivalence of the kind we have investigated may be a useful technique step toward the representation of knowledge.

## Acknowledgments:

Professor Yuzuru Tanaka of Hokkaido University, the inventor of the IntelligentPad system, participated extensively in the discussion and architectural planning of the ZipEditor. Tatsuya Kainuma of Fujitsu Aichi Engineering Limited implemented the version of the IntelligentPad system used in the ZipEditor and taught us how to use it. The Sapporo Electronic Center Foundation and its staff have provided a substantial research environment. We would also like to thank our corporate sponsors, in alphabetical order: ASCII Corporation, Fuji Xerox Co. Limited, Fujitsu Limited, Hitachi Limited, Nippon Telegraph and Telephone Corporation, and Tecnova Corporation. Finally, we extend special thanks to Project Xanadu, Mallicoat Enterprises and Vision Corporation.







## References

- [1] T.Nelson, "A File Structure for the Complex, the Changing and the Indeterminate", Proceedings of the ACM 20th National Conference, 1965.
- [2] T.Nelson, *Literary Machines 87.1*. (Latest edition: *Literary Machines 93.1*). Mindful Press, 3020 Bridgeway #295, Sausalito CA 94965.
- [3] Theodor Holm Nelson "Transcopyright: Pre-Permission for Virtual Republishing", available from the author [to be on Web by time of conference].
- [4] Theodor Holm Nelson, "A Publishing and Royalty Model for Networked Documents", *Proceedings: Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment*, 1995.
- [5] Theodor Holm Nelson, *Computer Lib*, Microsoft Edition, 1987.
- [6] Searching *hyperlink* from Alta Vista [Web address?], about 10000 matches were found on January 24, 1996.
- [7] Searching *transclusion* from Alta Vista (above), 79 matches were found on January 24, 1996.
- [8] The IntelligentPad System Page of Hokkaido University.
- [9] Fabio Vitali, David G. Durand. "Using versioning to support collaboration on the WWW," *Proceedings of the Fourth International WWW Conference '95, Boston, Massachusetts, USA, December 11-14, 1995*.
- [10] Terje Norderhaug, Juliet M. Oberding. "Designing a Web of Intellectual Property," *Proceedings of the Third International WWW Conference '95, Darmstadt, Germany, April 10-14, 1995*.







平成8年3月19日

## HtmlViewerPadの開発

日立製作所 中央研究所

日高 稔

### はじめに

WWW(ワールドワイドウェブ)と呼ばれる情報サーバーの普及の状況を見ると、このWWWを情報流通の基盤として利用しようとすることは自然であろう。IntelligentPadにおいて、Padの流通、Pad poolの構築は、ミームメディアとしての発展を促すものと考えられていることから[1]、札幌ハイパーラボでは、WWWをPad poolとし、Pad流通にWWWの通信機構を利用することを検討してきた。

WWWでは、情報提供の入口を、HomePageという形で構成しているが、Padの流通を促すためには、このHomePage上に直接Padを張り付けることが必要であると考えた。つまり、HomePageの構成要素としてImageが張り付けられているのと同様にPadも張り付けられるようにするということである。そうすることで、Padの提供者は作成したPadを簡単に広告することができ、Padのユーザーは、そのPadを簡単に試用することができる。

今回は、HTML文書に対してPadを張り付ける為の拡張と、拡張したHTMLを表示するためのViewerとして開発したHtmlViewerPadについて報告する。

### HtmlViewerPad

HTML(Hyper Text Markup Language)とは、SGML(Standard Generalized Markup Language)のサブセットとして構成されたMarkup Languageである。

Markup Languageとは、簡単にいえば、文書を表示する際に必要な文書のTextと文書の構成情報とをTagと呼ばれるMarkで分離し、実際に表示を行なうアプリケーションに対して独立に記述できるようにしたものである[2]。







そのため HTML 文書を表示するアプリケーションが実際に HTML を表示するためには、HTML 文書中の Tag を Parsing することと、Parsing 結果を表示可能な形に Format することが必要である。

そこで、基本的な機能は、NCSA Mosaic for X Window System のソースを利用することとし、今回はこれに手を加えて、Pad の利点を活かせるように拡張することにした。

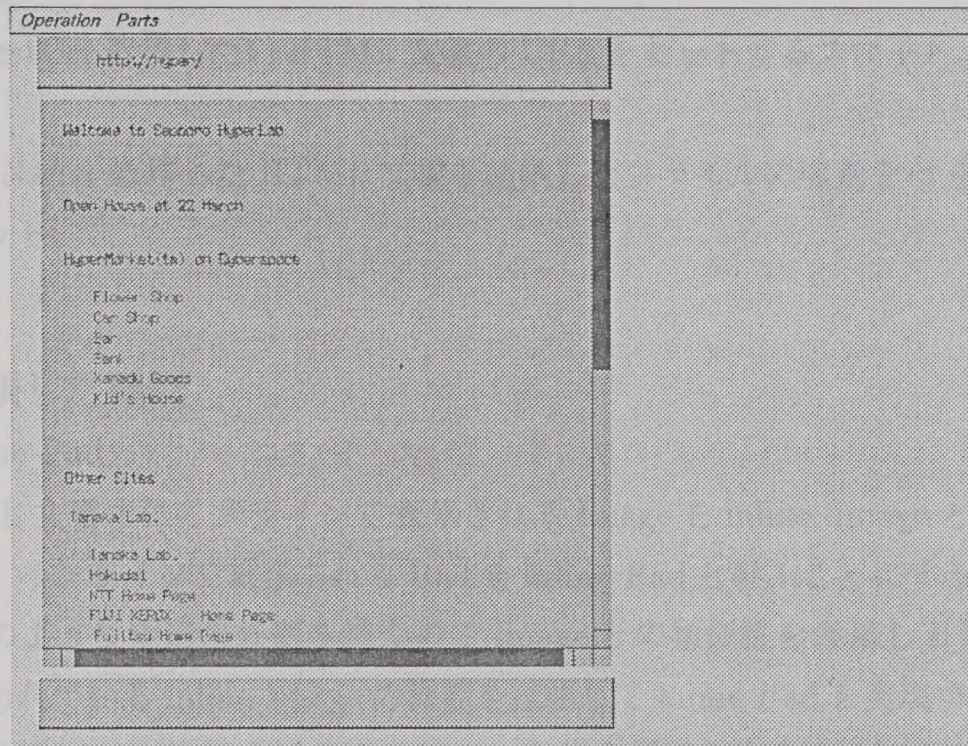


図 1 HtmlViewerPad

## HtmlViewerPad の仕様

HtmlViewerPad では、他の Pad とのインターフェイスとして次の 4 つのスロットを用意した。

### Slots

CurrentUrl 表示する HTML 文書の URL をセットするスロット。

NextUrl 表示している HTML 文書中にあるアンカーをマウスでポイントするとそのアンカーがセットされるスロット。

FileName 表示する HTML 文書を格納したファイルの名前をセットするスロット。

TextData 表示する HTML 文書のデータ自身をセットするスロット。







また、HtmlViewerPad は、URL で示されたファイルをリモートマシンにある WWW のサーバー (httpd) から受けとるために、サーバーとの通信機能を持った HttpPad をシステム上に一枚だけ作成する。

HttpPad は、次の 2 つのスロットにより、HtmlViewerPad と通信する。

Slots

RequestUrl 取得したい HTML 文書の URL をセットするスロット。

FileName 取得した HTML 文書を格納したファイルの名前をセットするスロット。

## 拡張した機能

inline Pad

HTML 文書中にインラインで表示される Image を inline Image と呼ぶように、インラインで表示される Pad を inline Pad と呼ぶことにする。Pad は紙のような矩形の外観を持つので Image との類似性を考慮し HTML で行なわれている inline Image の機能を拡張して inline Pad を実現する。

HTML では、文書中に inline Image を張る記述を次のように規定している。

```
< IMG SRC = "http://hostname/directory/filename.ext" >
```

inline Pad も上記の記述をそのまま利用しファイルネームの拡張子で Pad として認識するように拡張した。

inline Pad を HTML 文書中に張る記述は次のようになる。

```
< IMG SRC = "http://hostname/directory/filename.PAD" >
```

inline HTEXT

Netscape 社が行なっている HTML の拡張に FRAME 機能があるように、HTML 文書の構成要素として他の HTML 文書を用いることを望んでいる人は少なくないであろう。

HtmlViewerPad でもこの点を考慮して、HTML 文書の構成要素として HTML 文書を用いることができるようにした。







この機能も inline Pad と同様に HTML 文書中での記述は

```
<IMG SRC = "http://hostname/directory/filename.HTML" >
```

とし、ファイル名の拡張子により HTML 文書であることを認識する。

この時、元の HtmlViewerPad が表示する HTML 文書中に、inline HTEXT として張り付けられる実体は子 Pad としてジェネレートする HtmlViewerPad である。つまり、HtmlViewerPad が入れ子構造になる。

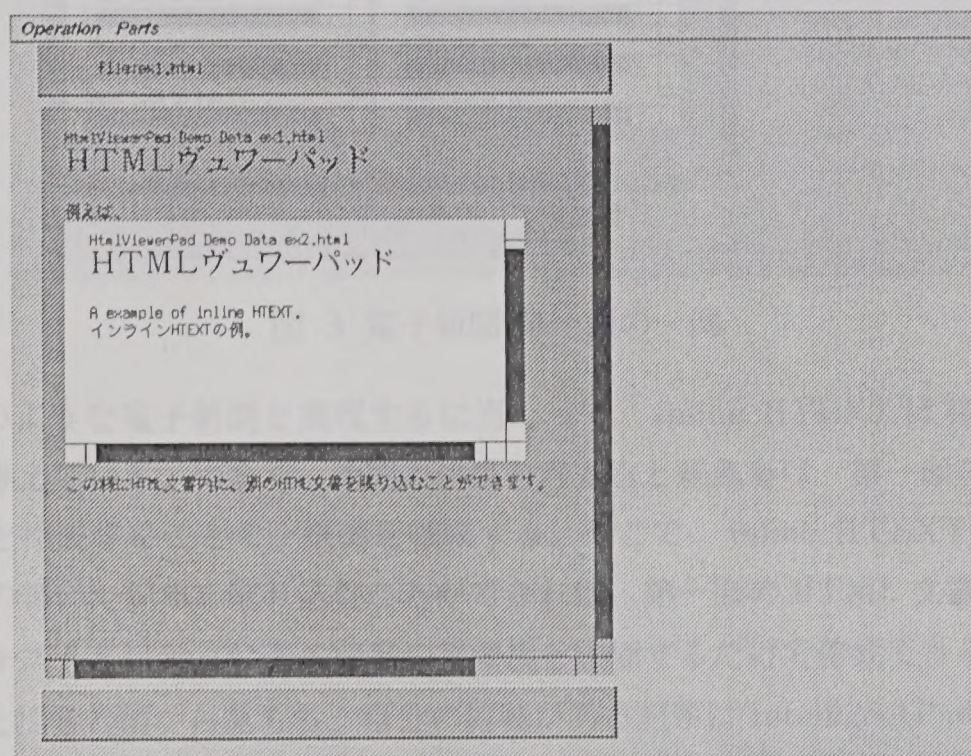


図 2 inline HTEXT

## HtmlViewerPad の応用例

現在検討している HtmlViewerPad の応用例として、電子新聞がある。新聞の構造は大まかにいって、複数の記者が書く記事とそれを適宜編集した紙面からなる。

従来の新聞では紙面の制約から編集者の判断した記事の重要度により見出しの有無、あるいは実際に掲載される紙面上の大きさなどの違いがあった。それは、読者側からは見ると情報を制限されているともいえる。

電子新聞でも同様に、最初に配送される第一面は記事の重要度により編集者の意図した紙面構成が行なわれるであろう。

しかし、ネットワーク上で配送されることを考えた時、料金、時間の制限を別にして、送付できる情報量に紙面等の制限はないため、最終的に得られる記事の量は読者の望む量となる。







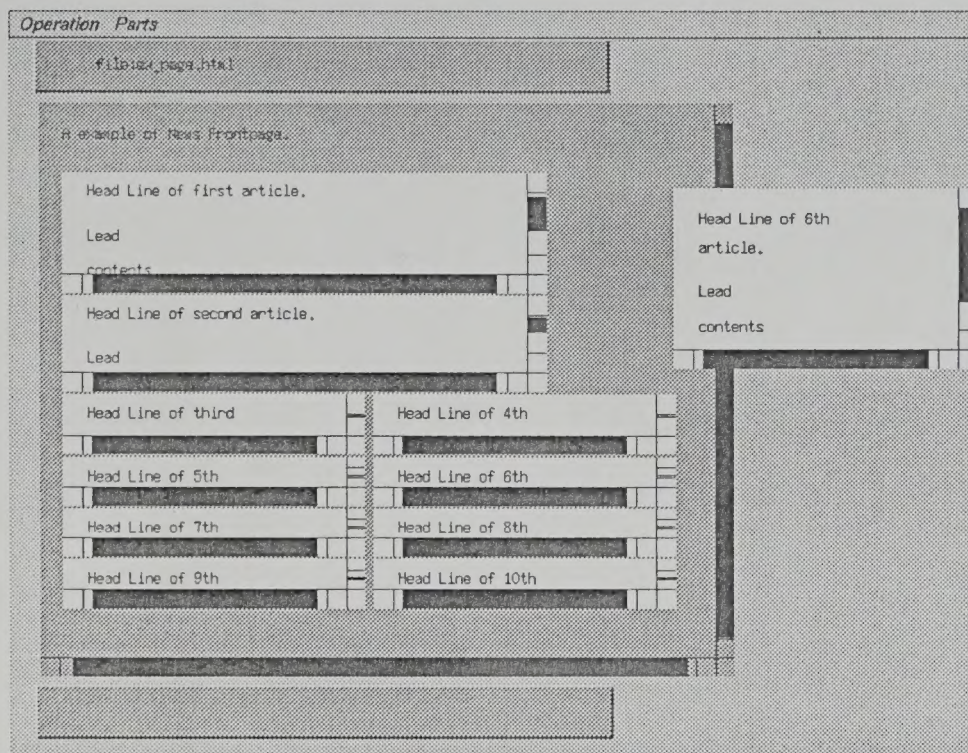


図 3 電子新聞 第一面の一例

このような電子新聞を実現するに当たって、inline HTEXTは有効であると考えている。つまり、新聞の構成を考えると編集者は、第一面のHTML文書を作成することで、紙面を構成する。そこで、inline HTEXTとして記者の書いた記事を取り込むことができれば、第一面のHTML文書は記事のテキストとは独立にその記事の重要度を評価するだけで作成できる。

また読者の側から見ても、自分の興味がある記事はIntelligentPadの持つ、Padの再構成可能という特徴を使って、簡単に自分の環境に取り出すことができ、適当にスクラップすることもできる。また簡単なユーザーインターフェイスで操作ができることも有用であろう。

### さいごに

今回開発したHtmlViewerPadは、それ自体ではHTML文書を表示することしかできないが、IntelligentPadが本来持っているさまざまな機能を簡単に流通できる機能を提供するという意味で有効であると考えられる。今後、IPadの機能をいかしたHomePageが作成されるようになれば、当初から意図している情報流通機構のクライアントアプリケーションの中核を担う部品となると考える。

### 参考文献

- [1]IntelligentPadと超流通, 田中譲, 信学技報 94-16.
- [2] 実践 SGML, Herwijnen, SGML 懇談会実用化 WG 監訳, 日本規格協会.















2. Pad の Agents 機能

# Network における Agents 機能

Sapporo HyperLab  
武部 義彦

## 1. 概要

従来の Network サービスでは、利用者がリクエストを出し、ホストがそれに答えることを繰り返すことで実現してきた。ホストが利用者のリクエストを受付結果を返すまで利用者は、次の処理を行うために待ち続ける必要があった。また、トラフィックが増加すると使用者の待ち時間も増加した。図1に従来の Network モデルを示す。

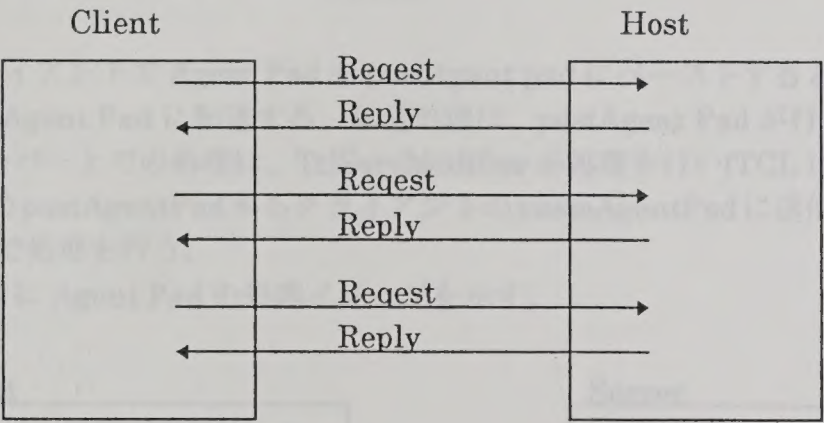


図1 従来の Network モデル

IntelligentPad の Network モデルでは、Pad 自身にプログラム(TCL)を持ち、その Pad を Server に転送し、Server でプログラムを実行する。実行結果を含む Pad を Client に転送することで Agent 機能を実現している。

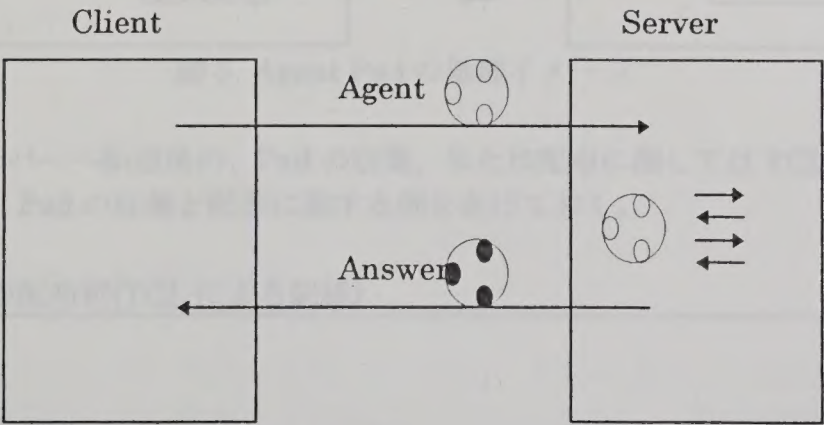


図2 IntelligentPad での Agent モデル







## 2. Pad の Agents 機能

Agents の処理は、TCL で記述する。作成した TCL は、TclServModifier にファイル名を指定することで使用可能となる。

### <変更点の特長>

- ・ 前回作成した Agents 機能は IntelligentPad の root に直接送信するため、送られてきた Pad であるかどうかの判別がつきにくかった。

今回作成した Agents 機能では、受信者が、受信用の Pad をロードしておくことで Agents 機能を持った Pad を送信するので、送信された Pad であるかの判定が行いやすい。

- ・ Command ボタンをなくすことで送信するデータ量が減少した。また、Command ボタンを配置する操作の必要がなくなった。

TclServModifier への TCL ファイルの設定方法を 5.1 に示す。

### ドラッグ&ドロップによる転送機能

クライアントで Agent Pad を postAgent pad にペーストするとサーバー側の pasteAgent Pad に転送する。転送処理は、postAgent Pad が行っている。

サーバー上での処理は、TclServModifier が処理を行い(TCL による処理)、サーバー側の postAgentPad からクライアントの pasteAgentPad に送信する処理についても TCL で処理を行う。

図 3 に Agent Pad の処理イメージを示す。

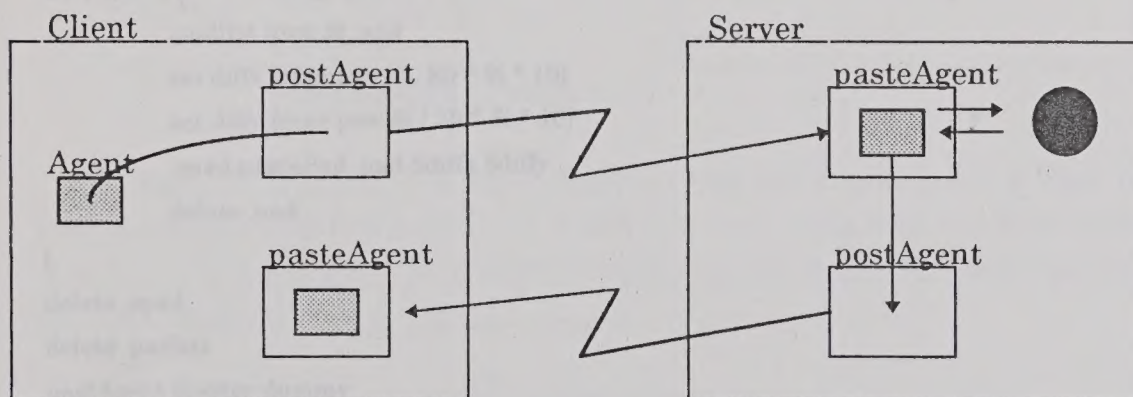


図 3 Agent Pad の処理イメージ

サーバーへ転送後の、Pad の収集、または配布に関しては TCL による記述が必要である。Pad の収集と配布に関する例をあげておく。

### Pad の配布例(TCL による記述)

```
createSlot button
proc s_button {} {
    query .padlist .rootCap "Stage.demo" "" ""
```







```

set count_ [padlist count]
print $count_
for {set i 0} {$i < $count_} {incr i 1}
    .padlist item $i .pad
    set diffx [expr pow(-1, $i) * $i 10]
    set diffy [expr pow(-1, [expr $i / 2]) * $i * 10]
    pastePad .pad $diffx $diffy
    delete .pad
}
delete .padlist
postAgent client Redsearch
vanish
}

proc Redsearch {} {
    query .padlist .rootcap "Stage" "" ""
    set count_ [padlist count]
    print $count_
    for {i 0} {$i < count_} {incr i 1} {
        .padlist item $i .spad
    }
    delete .padlist

    query .padlist .ThisCap "Red" "" ""
    set count_ [padlist count]
    print $count_
    for {set i 0} {$i < $count_} {incr i 1} {
        .padlist item $i .pad
        set diffx [expr pow(-1, $i) * $i * 10]
        set diffy [expr pow $i / 2] * $i * 10]
        .spad pastePad .pad $diffx $diffy
        delete .pad
    }
    delete .spad
    delete .padlist
    postAgent master dummy
}

proc dummy {} {}

```

### 3. 封筒 Pad による配送機能

特定のユーザに Pad を送信したい場合、または送信した Server の他のユーザから Pad を隠蔽したい場合、封筒のように指定されたユーザのみが開封できるような仕組みが必要である。







この封筒の概念を利用して封筒(Envelope) Pad の試作を行った。  
図4に封筒 Pad の構成を示す。

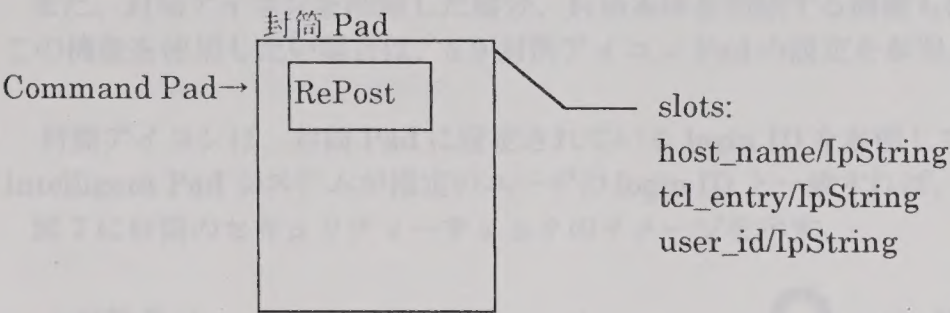


図4 封筒 Pad の構成

封筒を開封するような仕組みとして封筒アイコンを用意した。封筒アイコンの上でマウスの中ボタンを押すと封筒の実態が表示される。利用者はこのとき初めて封筒上にある Pad に対して copy や save の操作やスロット結合を行うことができる。

再度、封筒アイコン上でマウスの中ボタンを押すと封筒 Pad の実体が画面から見えなくなる。

図5に封筒 Pad の処理イメージを示し、封筒アイコンの作成方法を5.8に示す。

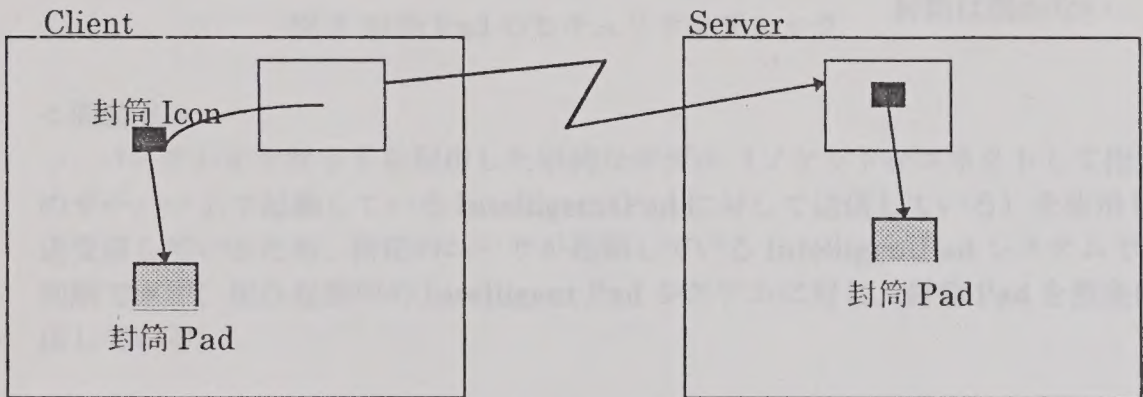


図5 封筒 Pad の処理イメージ

封筒 Pad 上に封筒 Pad が含まれる場合には封筒 Pad の再転送を封筒 Pad 上に Command Pad を張り合わせることで実現している。封筒 Pad 上に複数の封筒 Pad が含まれる場合も考慮し、一度の操作で封筒 Pad 上にある複数の封筒 Pad を別々の Server に転送することができる。

図6に Command ボタンを押した場合の同時転送のイメージを示す。

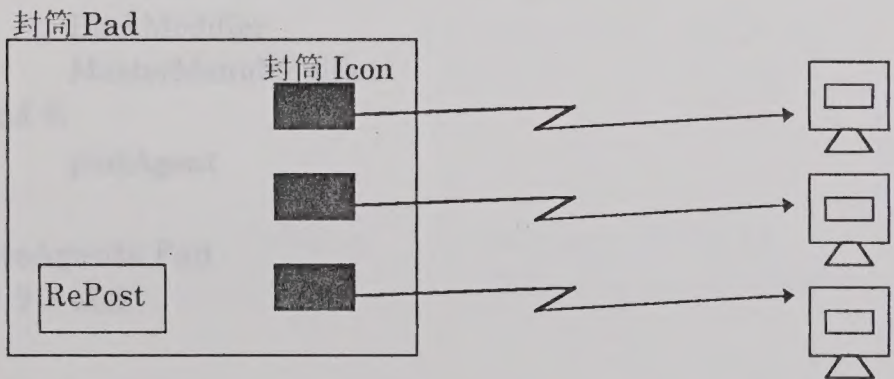








図6 同時転送のイメージ

また、封筒アイコンを削除した場合、封筒本体を削除する機能も提供する。  
この機能を使用したい場合は、5.9 封筒アイコン Pad の設定を参照。

封筒アイコンは、封筒 Pad に設定されている login ID を参照して、起動中の Intelligent Pad システムが指定のユーザの login ID と一致すれば、封筒を開封する。  
図7に封筒のセキュリティーチェックのイメージを示す。

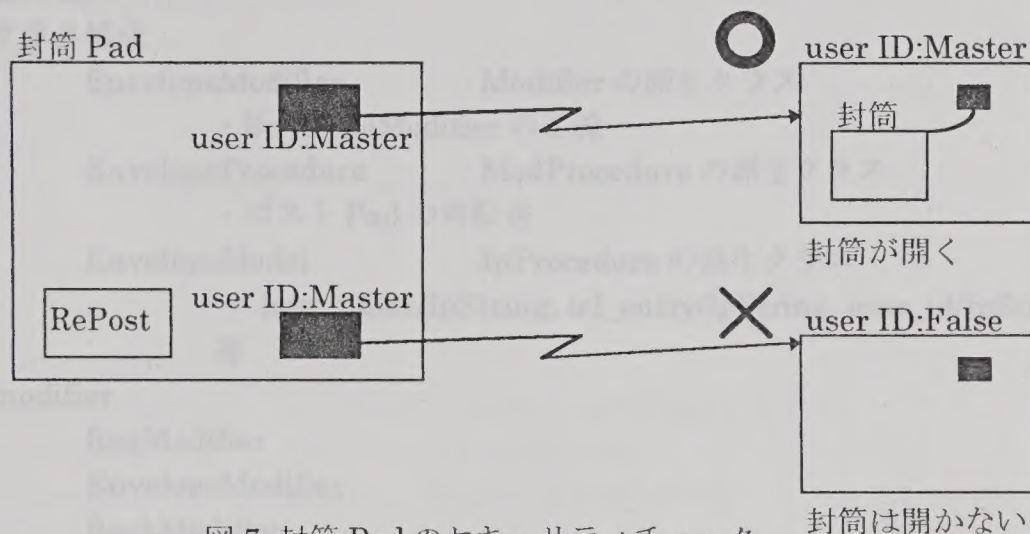


図7 封筒 Pad のセキュリティーチェック

#### <問題点>

- ・ バックレイソケットを利用した単純なモデル（ソケットをコネクして指定したのサーバー上で起動している IntelligentntPad に対して送信している）を使用して、送受信しているため、指定のユーザが起動している IntelligentPad システムであるか判断できず、現在起動中の Intelligent Pad システムに対し、封筒 Pad を無条件に送信している。

## 4. クラス定義

### postAgent Pad

#### ・クラス構成

postAgentsCapsule : Capsule の派生クラス  
paste イベントの処理

#### ・modifier

ReqModifier  
TIFFModifier  
BackModifier  
MasterMenuModifier

#### ・Pad 名

postAgent

### pasteAgents Pad

#### ・クラス構成







- modifier

- ReqModifier
  - TIFFModifier
  - BackModifier
  - MasterMenuModifier

- Pad 名

- pasteAgent

### 封筒 Pad

- クラス構成

- EnvelopeModifier : Modifier の派生クラス

- EnvelopeModifier の定義

- EnvelopeProcedure : ModProcedure の派生クラス

- ポスト Pad の再転送

- EnvelopeModel : IpProcedure の派生クラス

- host\_name/IpString, tcl\_entry/IpString, user\_id/IpString の定義

- modifier

- ReqModifier
  - EnvelopeModifier
  - BackModifier
  - MasterMenuModifier

- Pad 名

- envelope

- slot

- host\_name/Ipstring : "host\_name"
  - tcl\_entry/IpString : "TCL entry name"
  - user\_id/IpString : "none"

### ポスト Pad

- クラス構成

- postCapsule : Capsule の派生クラス

- paste イベントの処理

- Pad の転送

- postProcedure : IpProcedure の派生クラス

- modifier

- ReqModifier
  - TIFFModifier
  - BackModifier
  - MasterMenuModifier

- pad 名

- post

- slot

- host\_name/IpString : "host\_name"(不要?)
  - tcl\_entry/IpString : "TCL entry name" (不要?)







## 郵便箱 Pad

- ・ クラス構成
  - pasteCapsule : Capsule
  - pasteProcedure : IpProcedure
- ・ modifier
  - ReqModifier
  - TIFFModifier
  - BackModifier
  - MasterMenuModifier
- ・ pad 名
  - paste
- ・ slot
  - label/IpString : “Command” (未使用)

## 5. 操作

### 5.1 TCL ファイルの設定方法

IntelligentPad のインストールディレクトリを”EvsSys”とする。

- ① “EvsSys/tool/tcl/”に TCL ファイルを作成する。
- ② Operation の Load を選択し、Load 一覧を表示する。
- ③ “agent/”を選択する。
- ④ “agent.pad”を選択する。
- ⑤ マウスカーソルをロードされた Pad に移動する。
- ⑥ マウスの右ボタンを押し、メニューを表示する。
- ⑦ Slave Menu を選択し、メニューを表示する。
- ⑧ “Modifier Sheet”を選択する。
- ⑨ “TclServModifier”を取り出す。
- ⑩ ①～⑧までの動作と同様に Agent 用の Pad をロードし、Modifier Sheet を表示する。
- ⑪ “TclServModifier”が 1 となるような場所に挿入する。

### 5.2 ポスト Pad のロード

- ① “Parts”の”User Parts”を選択する。
- ② User Parts ディレクトリ一覧から”envelope/”を選択する。
- ③ “post”を選択する。

### 5.3 郵便箱 Pad のロード

- ① “Parts”の”User Parts”を選択する。
- ② User Parts のディレクトリ一覧から”envelope”を選択する。
- ③ “paste”を選択する。

### 5.4 封筒 Pad のロード

- ① “Parts”の”User Parts”を選択する。
- ② User parts のディレクトリ一覧から”envelope”を選択する。
- ③ “envelope”を選択する。





### 5.5 封筒 Pad のサイズ変更

- ① マウスカーソルを封筒 Pad 上に移動する。
- ② マウスの右ボタンを押し、メニューを表示する。
- ③ “Select modifier”を選択し、メニューを表示する。
- ④ “Resize/Move Frame”を選択する。
- ⑤ マウスカーソルを封筒 Pad 上に移動する。
- ⑥ マウスの右ボタンを押しながら移動する。

### 5.6 Pad の状態復帰

- ① マウスカーソルを封筒 Pad 上に移動する。
- ② マウスの右ボタンを押し、メニューを表示する。
- ③ “Select modifier”を選択し、メニューを表示する。
- ④ “CapGlyph”を選択する。

### 5.7 封筒 Pad の作成

#### Command Pad の貼り付け

- ① “Parts”の”Basic Parts”を選択する。
- ② Basic Parts 一覧から“Command”を選択する。
- ③ 封筒 Pad 上の任意の位置に貼り付ける。
- ④ マウスカーソルを Command Pad”の周辺に移動する。
- ⑤ マウスの右ボタンを押し、メニューを表示する。
- ⑥ “Slave Menu”を選択し、メニューを表示する。
- ⑦ “Vconnect”を選択し、Slot 一覧を表示する。
- ⑧ “button/Command”を選択する。このとき、Connect To 一覧の状態は、“set”、“Update”、“Gimme”とも変更する必要はない。

#### 送信先の設定

- ① “Parts”の”Basic Parts”を選択する。
- ② Basic Parts 一覧から”Text”を選択する。
- ③ マウスカーソルを Text Pad 上に移動する。
- ④ マウスの右ボタンを押し、メニューを表示する
- ⑤ “Slave Menu”を選択し、メニューを表示する
- ⑥ “Hconnect”を選択しする。
- ⑦ マウスカーソルを Text Pad 上に移動する。
- ⑧ マウスの右ボタンを押しながら、マウスを封筒 Pad 上に移動する。このとき、Text pad から移動中のマウスカーソルに対して線が表示される。
- ⑨ “host\_name/IpString”を選択する。このとき、Connect To 一覧の状態は、“set”、“Update”、“Gimme”とも変更する必要はない。
- ⑩ マウスカーソルを Text Pad 上に移動する。
- ⑪ テキストを受信者のマシン名、または IP アドレスに修正する。

#### Text Pad の削除

- ① マウスカーソルを Text Pad 上に移動する。
- ② マウスの右ボタンを押し、メニューを表示する。
- ③ “Delete”を選択する。





#### Tcl ファイルの指定

- ① “Parts”の”Basic Parts”を選択する。
- ② Basic Parts 一覧から”Text”を選択する。
- ③ マウ斯卡ーソルを Text Pad 上に移動する。
- ④ マウスの右ボタンを押し、メニューを表示する
- ⑤ “Slave Menu”を選択し、メニューを表示する
- ⑥ “Hconnect”を選択しする。
- ⑦ マウ斯卡ーソルを Text Pad 上に移動する。
- ⑧ マウスの右ボタンを押しながら、マウスを封筒 Pad 上に移動する。このとき、Text pad から移動中のマウ斯卡ーソルに対して線が表示される。
- ⑨ “tcl\_entry/IpString”を選択する。このとき、Connect To 一覧の状態は、”set”、”Update”、“Gimme”とも変更する必要はない。
- ⑩ マウ斯卡ーソルを Text Pad 上に移動する。
- ⑪ テキストを Tcl のファイル名に修正する。

#### ユーザ ID の指定

- ① “Parts”の”Basic Parts”を選択する。
- ② Basic Parts 一覧から”Text”を選択する。
- ③ マウ斯卡ーソルを Text Pad 上に移動する。
- ④ マウスの右ボタンを押し、メニューを表示する
- ⑤ “Slave Menu”を選択し、メニューを表示する
- ⑥ “Hconnect”を選択しする。
- ⑦ マウ斯卡ーソルを Text Pad 上に移動する。
- ⑧ マウスの右ボタンを押しながら、マウスを封筒 Pad 上に移動する。このとき、Text pad から移動中のマウ斯卡ーソルに対して線が表示される。
- ⑨ “user\_id/IpString”を選択する。このとき、Connect To 一覧の状態は、”set”、”Update”、“Gimme”とも変更する必要はない。
- ⑩ マウ斯卡ーソルを Text Pad 上に移動する。
- ⑪ テキストを受信者のユーザ ID に修正する。

封筒 Pad の雛形は、”load”の”envelope”に”envelope”でセーブされている。

#### 5.8 封筒アイコン Pad の作成

- ① 5.7 封筒 Pad の作成手順に従い、封筒 Pad を作成しておく。
- ② マウ斯卡ーソルを封筒 Pad 上に移動する。
- ③ マウスの右ボタンを押し、メニューを表示する。
- ④ “Slave Menu”を選択し、メニューを表示する。
- ⑤ “Shrink”を選択すると、封筒アイコン Pad が表示される。

#### 5.9 封筒アイコン Pad の設定(封筒アイコン Pad を削除すると封筒 Pad を同時削除)

- ① 5.8 封筒アイコン Pad の作成に従い、封筒アイコン Pad を作成しておく。
- ② 封筒アイコン Pad にマウ斯卡ーソルを移動する。
- ③ マウスの右ボタンを押し、メニューを表示する。
- ④ “Slave menu”を選択し、メニューを表示する。
- ⑤ “Modifier Sheet”を選択し、Modifier Sheet Pad を表示する。
- ⑥ マウ斯卡ーソルを”4: IconicAnchorModifier”に移動する。





- ⑦ “shift”キーを押しながら、マウスの右ボタンを押す。
- ⑧ “DelMaster”の項目をチェックする。
- ⑨ ‘quit’を押す。

## 6. 考察 (Pad の課金/流通)

Evasys を利用した Pad の流通

図 8 に Pad の流れを示す。

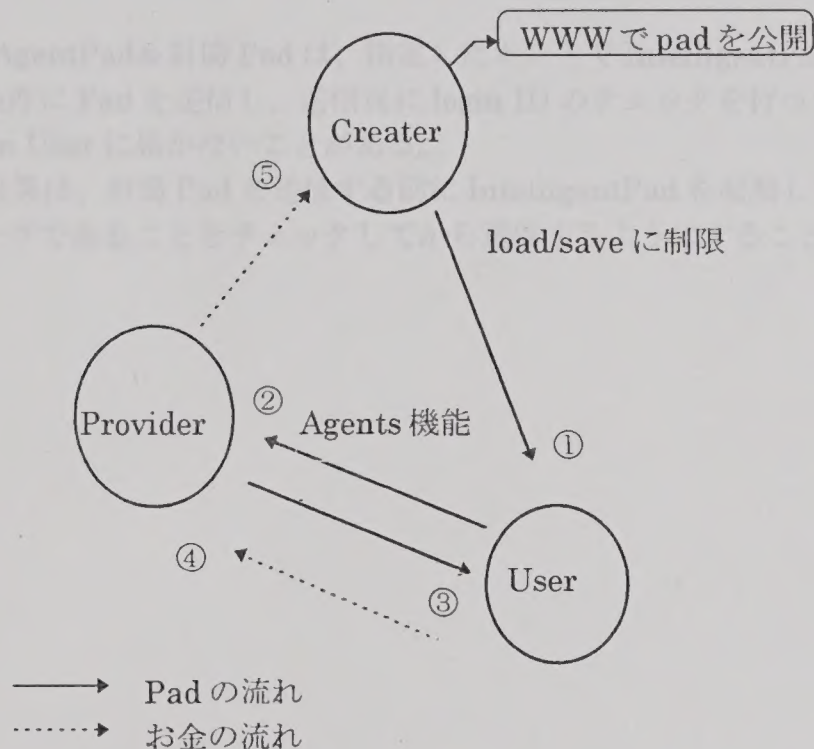


図 8 Pad の流れ

簡易タイプ：

- ① Pad のロード、セーブなどに利用制限を付ける。  
(Pad に login 名を持つ slot を作成し、load/save など現在 login 名と比較する。)
- ② Agents の機能を使って転送する。(このときホスト名、login user ID を渡す)
- ③ Pad の slot 名を変更し、再転送する。
- ④、⑤ pending  
(個人的には Digicash/SyberCash などのメジャーなものを利用するのがよいのではと考えている)

(問題点と対策)

- ・ ①で利用制限を付けても User が既に持っている Pad の上に paste を行い、save した場合は、子 Pad として Save されてしまう。  
→ Pad 自身にレベルを持ち、自分自身のレベルより、低いレベルの Pad 上に Past できないようにする。
- ・ login ID によるチェックであるため、別 site の同一 login ID 上で動作する。





→システムインストール時に暗号化されたユニークな password を作成し、password によるチェックを行う。

## 7. 課題

- ・ 転送後の Pad の位置がうまく設定できていない。これは、Pad 内に保持している座標が左下を基準とした絶対座標であることが影響している。これは、親 Pad からの相対座標を保持することで解決する。

- ・ AgentPad&封筒 Pad は、指定したホストで IntelligentPad が起動されていれば、無条件に Pad を送信し、送信後に login ID のチェックを行っているため、指定した login User に届かないことがある。

対策は、封筒 Pad を送信する前に IntelligentPad を起動しているユーザが指定のユーザであることをチェックしてから送信するようにすることで解決できる。

— 以上 —











# Visible Connections: The Next Step in Electronic Media

*The HyperLab Zip Editor as Prototype  
for Transparallel Media  
and Work Systems*

----- Theodor Holm Nelson  
Hokkaido University  
and  
Sapporo HyperLab  
e-mail: ted@xanadu.net -----

Keywords: \* TRANSPARENT \* HYPERTEXT \* HYPERMEDIUM \* HYPERLAB \*

HYPERTEXT

# Visible Connections: The Next Step in Electronic Media

The HyperLab Zip Editor as Prototype  
for Transparallel Media  
and Work Systems

Thomas H. Dyer  
University of Illinois at Chicago  
Chicago, Illinois 60607  
USA  
tdyer@uic.edu



# PROBLEMS OF THE INFORMATION WORLD I: THE HOPE AND DIFFICULTY OF COMPUTERS

**Summary.** This paper discusses a new type of electronic medium and system of work (originally proposed by the author in 1965), now implemented on workstations (and with an interface to the World Wide Web). While this design has been in print for thirtyfive years, from the article that introduced the idea of hypertext, it has not been implemented until now. The original 1965 article which introduced the idea of hypertext discussed not only links but transclusion (though the term was introduced later), and presented a unified design which has not been implemented until the present time. This paper discusses various design aspects of such systems, called xanalogical systems.

Such a system has now been implemented at Sapporo HyperLab, and we discuss various aspects of the design and implementation. It offers both links (as in conventional hypermedia) and transclusion, or instancing in contexts which may be compared. The current implementation is built around IntelligentPad as a working interface, but has functions that can be used through World Wide Web browsers. Its most unusual aspect may be the implementation of hyperarchical outlining, allowing several different hierarchies to be maintained and intercompared simultaneously.

Keywords: • TRANSCLUSION • TRANSPARALLEL MEDIA •  
HYPERTEXT

**Summary:** This paper discusses a new type of electronic medium and system of text originally proposed by the author in 1965, now implemented as *World Wide Web*. While this design has been in print for thirty years, from the article that introduced the idea of hypertext, it has not been implemented until now. The original 1965 article which introduced the idea of hypertext discussed not only links but transclusion (though the term was introduced later), and presented a unified design which has not been implemented until the present time. This paper discusses various design aspects of such systems, called *analogical systems*.

Such a system has now been implemented at *Stapton* Hypertext, and we discuss various aspects of the design and implementation. It offers both links (as in conventional hypertext) and transclusion, or including in contexts which may be compared. The current implementation is built around intelligent as a working interface, but has functions that can be used through *World Wide Web* browser. Its most unusual aspect may be the implementation of hypertextual outlining, allowing several different hierarchies to be maintained and intercompared simultaneously.

**Keywords:** TRANSCLUSSION • TRANSPARENT MEDIA • HYPERTEXT



# PROBLEMS OF THE INFORMATION WORLD 1: THE HOPE AND DIFFICULTY OF COMPUTERS

The great hope of the computer age is for improved understanding, visualization and creative control of our work and our lives. This paper is an attempt to contribute to that goal.

But a crucial problem of today's information world is that information can be hard to understand. The problem is worse today than ever before for two reasons: first, the world is more complicated before; second, we realize that we must track and understand complexes of information more intricate than people have ever tried to understand before. We need to be able to see the relations among complex structures, with every kind of interconnection.

There are many complex structures and ideas in the world to see, understand, and share. How can we see and understand them? How can we share them?

It is my objective here to clarify one possible avenue to this seeing, sharing and understanding.

Computers should help, but there are many obstacles. Computers are good at storage and delivery, but do not easily help in understanding. Our common structures for handling information, such as data base and word processing, are not convenient for users and give only limited overview and understanding. These forms of presentation are not especially clear or understandable to users, especially when complicated structures need to be understood.

But since these are artificial constructs, why can't we find better constructs for these purposes? The artificial constructs of database and text systems, and their details of their different designs, were created by human beings in search of something useful.





We need much more. How can we see and understand the world of information best?

Database and text systems are only two among the many possible designs for information storage and presentation. And the possible designs of electronic media are completely arbitrary.

Digital media are not constrained by the kinds of physical restrictions that limit such traditional media as theatrical performances and books. The possible designs of such electronic media can be built around a great variety of concepts, and very few have been tried.

Digital media, unlike physical media (and even analog electronics), can have any kind of conceptual structure. Computer-screen media may have any kinds of windows and overlays; digital audio and video media may have any number of tracks and channels; computer interactive media may have any new forms of connection and forms of interactivity.

Thus the options of design are vast; but then so is the difficulty of choosing.

## **PROBLEMS OF THE INFORMATION WORLD 2: KINDS OF COMPLEXITY-- STRUCTURE, INTERCONNECTION AND KEEPING TRACK OF CHANGES.**

*Interconnection.* There are many types of interconnection. How can we show them all in a way that people can understand?

*Change.* Needing to understand change makes the problem even harder. There is so much to keep track of, especially changes in ongoing work. We need to keep track of many intersecting and changing lists. We need to keep track of constantly changing text documents. And this is especially true in today's hypertext and





“multimedia” contents, which often need to be changed constantly.

These problems all multiply together to make one big unified software problem: *how can we maintain connections, order and understanding among things which are constantly changing.?*

Neither paper nor computer methods do much for these problems yet.

## **PROBLEMS OF THE INFORMATION WORLD 3: THE NEED FOR INTERCOMPARISON AND RE-USE**

These problems are made worse by the need for intercomparison and the need for re-use.

*Intercomparison.* There is a special class of interconnections: those involving corresponding materials-- things which are matched one-to-one, things which are similar to each other, and things which share material. This means that we need to intercompare the correspondences of objects and structures; either because they are intrinsically connected side by side, or because there is some useful parallelism.

*Re-use.* We need to re-use many kinds of data, and to know where they came from, and what their original form was. (I see the copyright issue as the same problem. The problem of copyright may be seen as a problem of re-use. Indeed, knowing the origin of copyrighted material is perhaps more important conceptually than issues of royalty.)





## PROBLEMS OF THE INFORMATION WORLD 4: THE NEED TO CONNECT TO ELECTRONICALLY PUBLISHED MATERIALS

Now that electronic publication on networks has begun-- especially on the World Wide Web-- people need to make links to those published materials. People need to make private comments and private bookmarks, as well as to publish comments and annotations.

On World Wide Web, anyone may create simple hypertext links. But the limitations of these simple links are becoming clear to everyone: material may be connected only by untyped links, and the links may only be seen in one direction.

How should people be able to connect their media objects to those already published?





# STRUCTURES FOR UNDERSTANDING

I believe that these problems of understanding information, mentioned earlier, represent a fundamental issue of media design. We need to see and understand all the relations that exist in thought. Thus *we must design media* that satisfactorily represent, and present, all the relations that exist in thought.

But the problem of representing all the possible forms of thought could involve us in astronomical complexity. We want a system that can model, or represent the complexity of things as they are, but in a way that is comprehensible to the mind, and thus presumably simpler than things as they *really* are. So rather than get into a crazed system of representation for all thinking, it may be beneficial to cut the problem down somewhat. Naturally, we need to compress these relations into some manageable subset. (Everyone compresses ideas into some subset, even if it is just the sequential words and sentences of ordinary prose.) Therefore, what subset is the most powerful while still simple?

## **1. WE NEED STRUCTURES FOR UNDERSTANDING.**

What structures will best help our understanding?

## **2. WE NEED STRUCTURES FOR KEEPING TRACK.**

As a special example, what about keeping track of changing and interrelated things-- events, physical objects, published ideas? What structures will best help us keep track of such change?

## **3. WE NEED STRUCTURES FOR UNDERSTANDING ALTERNATIVES.**

And this becomes a work system problem: How can we handle alternative organizations of the same materials, and viewing and manipulating these alternatives?

This includes alternatives plans, alternative designs, alternative interpretations of facts. In all these cases, the same material takes on different but partially-corresponding form.

This problem of alternatives has an interesting variety. On the individual scale, complex alternatives are those about complex decisions and plans made by individuals and small groups. But the same issue confronts society as a whole on the megascale. Written literature as we have known it is a system that has helped







us consider different ways to organize things, and make complex decisions. Thus the societal question, "How should tomorrow's literature be defined"-- is very much the same problem of presenting complex alternatives-- for scholars and politicians, designers and artists, writers and businessmen. The problem has an odd similarity at all scales.

Such a media design for handling complex alternatives should be *hyperarchitectural*, that is, permitting the same material to be organized at the same time with simultaneous alternative sequences, hierarchies, structures. These would not be treated as contradictory but rather as *supercomplete*-- containing materials which may or may not clash.

Can it be possible that similar media designs could be found for all of these?

## THE MEANING OF "MEDIA"

The term "media" is used in a number of different ways, to mean a number of different things, without clarification-- especially, it seems, by those who use the term frequently.\*

\*At a recent conference I asked Nicholas Negroponte, head of MIT's Media Laboratory, to define the term "media." His response was rather perplexing: "Media is air," he said. He claimed to mean it literally.

To try to clarify some of these issues, let me try to distinguish some key senses of the term "media."

1. **STORAGE MEDIA** (paper, floppy disk).
2. **MEDIA CHANNELS** (speech, video, typography, printing, radio waves) -- technical systems or methods of representation, presentation and distribution.
3. **GENERIC MEDIA PACKAGES OR VESSELS** created using media channels: such specific abstract structures as "song,"





“poem,” “book,” “movie,” “news program,” “television show,” “novel”... These are recognizable units into which media contents may be organized.

**4. MEDIA CONTENTS:** the units which can be put into media packages-- words, footage, audio samples...

**5. MEDIA WORKS:** media contents which have been organized into recognizable media packages. (Characteristically, media works have points of view and can be censored.)

Finally,

**6. MEDIA WORK SYSTEMS** -- environments for the construction and reworking of media contents into media works. (Note that in earlier days, media work systems did not have to be designed; writing and composing could take place anywhere, under many different circumstances. But today's digital media need to be created in digital settings.)

When Marshall McLuhan spoke of “media,” he may have meant meant all of these senses taken together; or he may have meant particular combinations. It is hard to know, because his writing was rather mysterious. “The medium is the message,” McLuhan's famous saying, could perhaps be translated as: “Particular media channels and vessels carry a system of ideas that is independent of their contents and works.”

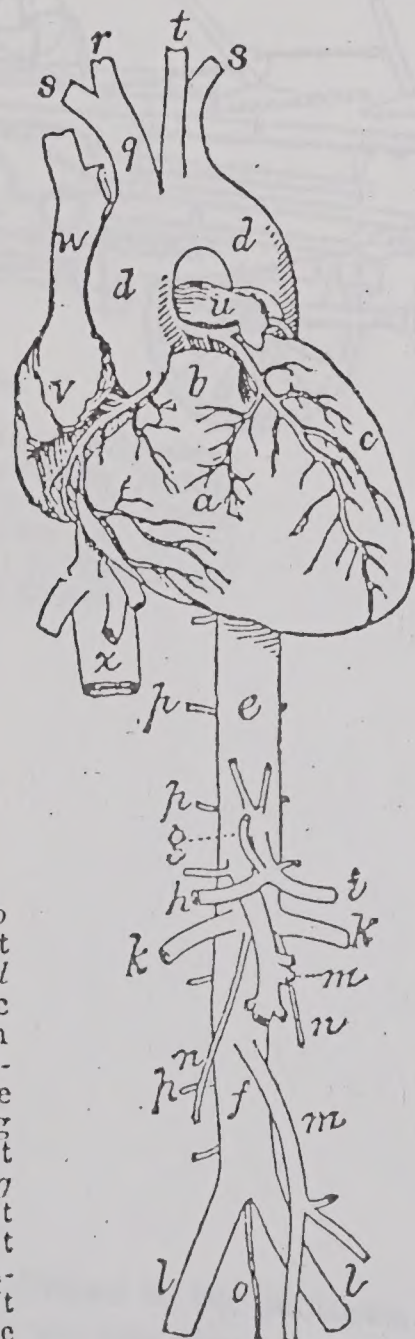
By making these distinctions, it is easier clarify my statement that the problem of understanding is a problem of media design. I am talking about the design of media channels, media packages and media work systems. I believe that such media channels, media packages and media work systems can be designed for the clarification of complex structure.





# TRANSPARALLEL MEDIA

We do not need to look very far to find inspiration for the design of such media. There is a tradition of media where detailed connections are explicitly shown, and it can be found in many places in the print world: as close as an ordinary dictionary. For example, consider these pictures: illustrations with detailed captions referring to their specific parts. (Examples from Webster's New International Dictionary, 1919 edition).

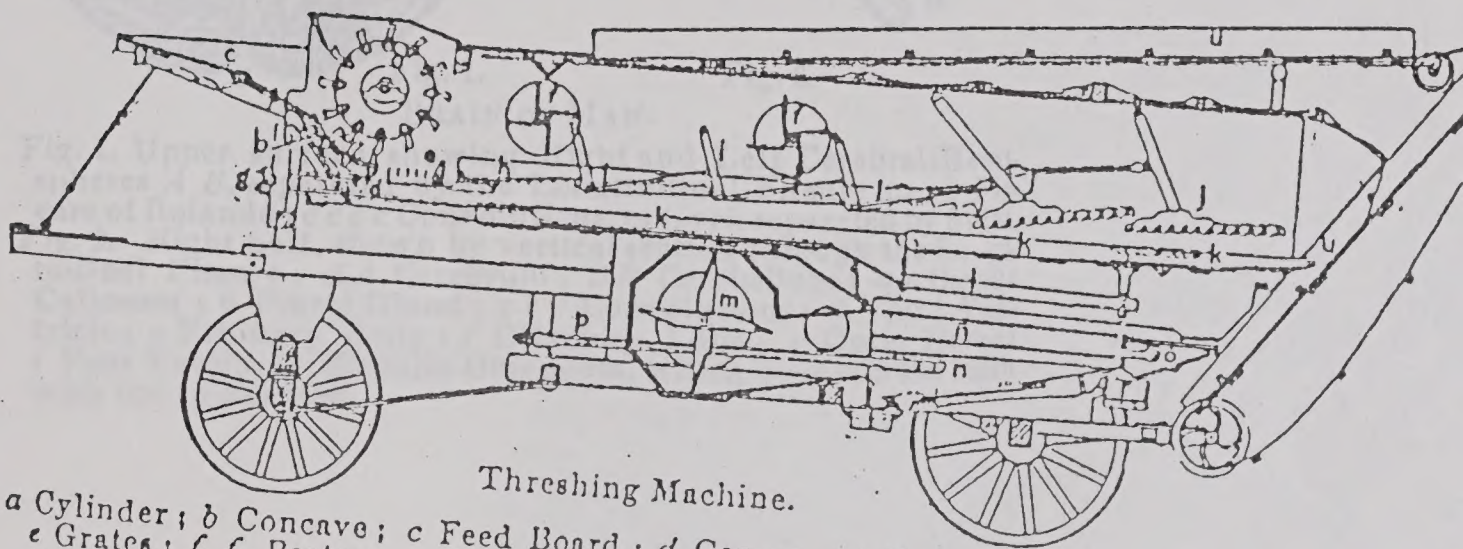


*Description of the Illustration:*  
a Right Ventricle of heart, with b Stump of Pulmonary Artery. c Left Ventricle, connecting with Aorta (dd Arch of Aorta, e Descending Thoracic Aorta, f Abdominal Aorta), which shows the Coronary Arteries branching from it over the surface of the heart, and the stumps of the following arteries: r Right Carotid, s s Right and Left Subclavian, t Left Carotid, g Gastric, h Hepatic, i Splenic, k k Right and Left Renal, l l Right and Left Common Iliac, m m Superior and Inferior Mesenteric, n n Right and Left Spermatic, o Middle Sacral, p p Some of the Intercostal and Lumbar Arteries q Innominate Artery.  
u Left Auricular Appendix. v Right Auricle and Appendix, receiving w Superior Vena Cava descending from Right and Left Innominate Veins, and x Inferior Vena Cava ascending from below, with stumps of Hepatic Veins.

Aorta and Heart of Man, front view.







Threshing Machine.

a Cylinder; b Concave; c Feed Board; d Concave Adjuster;  
e Grates; f, f, Beaters; g Crank and Pitmans driving Riddle  
(h, i, j) and Grain Pan (k, k); u, u Straw Stacker, with apron  
folded for transportation; l Distributor; m Fan for blast;  
n, n Shoe with Sieves and Screen; o Tail Shoe; p Crank and  
Pitman driving Shoe; r Tailing Spout; t Grain Spout.

In the above examples, the captions were linked to the pictures. In other common examples (same source), we can also see parallel pictures whose parts correspond:





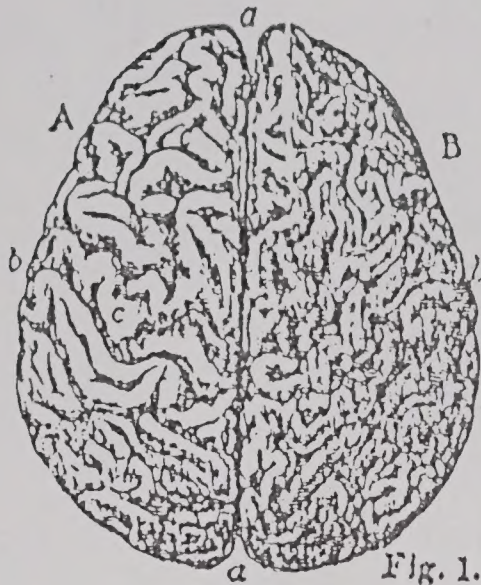


Fig. 1.

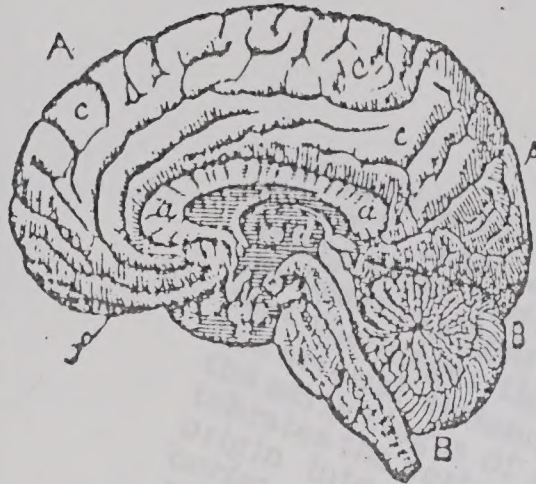
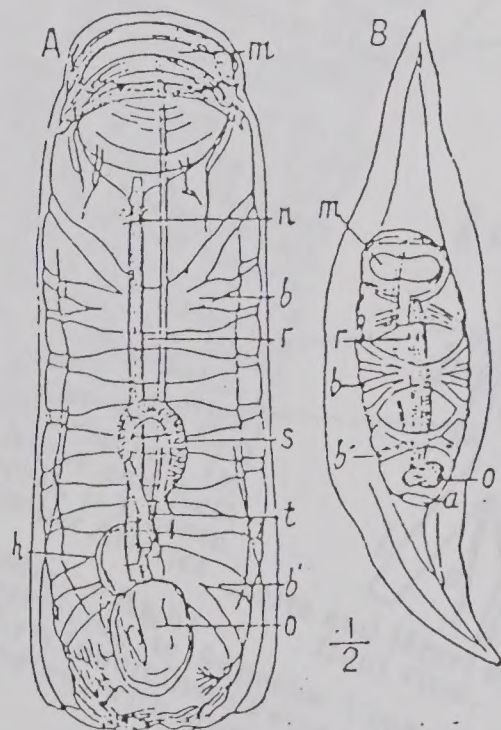


Fig. 2.

BRAIN OF MAN.

Fig. 1. Upper surface, showing Right and Left Cerebral Hemispheres *A B*, separated by the Longitudinal Fissure *aa*; *b* Fissure of Rolando; *c c c* Convolutions, or Gyri, separated by Sulci.  
Fig. 2. Right half, shown by vertical section through the Longitudinal Fissure; *AA* Cerebrum; *BB* Cerebellum; *aa* Corpus Callosum; *b* Pineal Gland; *c c c* Convolutions; *d* Third Ventricle; *e* Pituitary Body; *f* Olfactory Lobe; *g* Optic Nerve; *i* Pons Varolii; *k* Medulla Oblongata, which connects the brain with the spinal cord.



Salpa. *A* Solitary Individual. *B* Individual from a chain. *m* Branchial Aperture; *a* Atrial Aperture; *b b'* Muscular Bands; *n* Nerve Ganglion; *r* Dorsal Lamina; *h* Heart; *o* Nucleus; *s* Chain of young Zooids developing from the Stolon *t*.

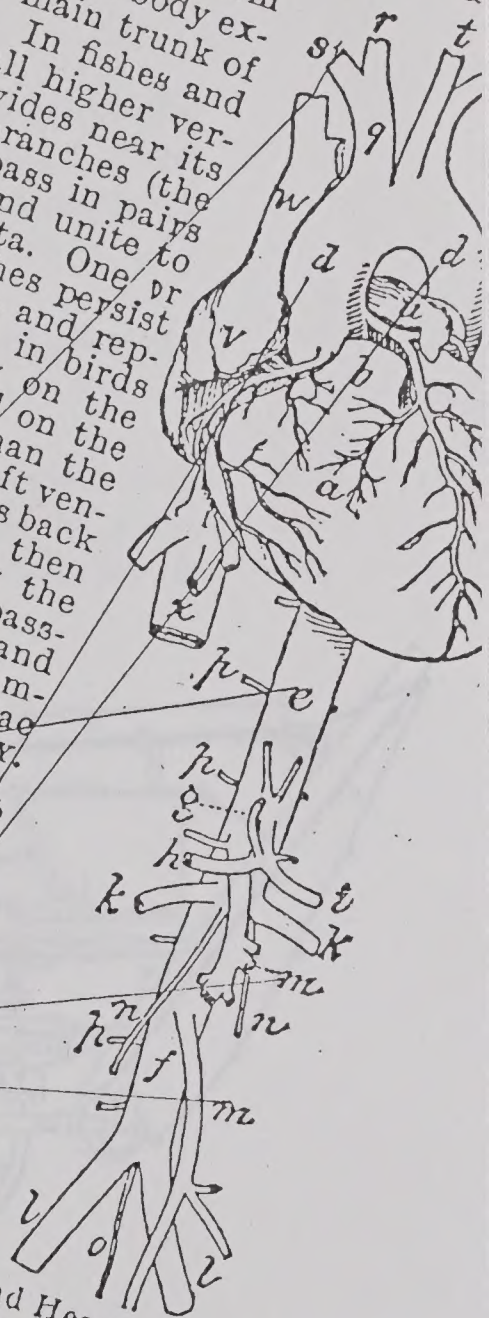




Note that lines can be added to show relationship of choice in visualization, since no new informa

**a-or'ta** (ā-ōr'tā), *n.* [NL., fr. Gr. ἀορτή, fr. ἀ- heave.] *Anal. & Zool.* The great artery which carries the blood from the heart to all parts of the body except the lungs; the main trunk of the arterial system. In fishes and the early stages of all higher vertebrates the aorta divides near its origin into several branches (the *aortic arches*) which pass in pairs round the esophagus and unite to form the systemic aorta. One or more pairs of these arches persist in the adult in amphibians and reptiles, but only one arch in birds and mammals, this being in the right side in the birds and on the left side in the mammals. In man the aorta extends up from the left ventricle of the heart and arches back over the root of the thorax along the left side of the spinal column, passing through the diaphragm, and dividing opposite the fourth lumbar vertebra into the common iliac arteries. See **INNOMINATE ARTERY**.

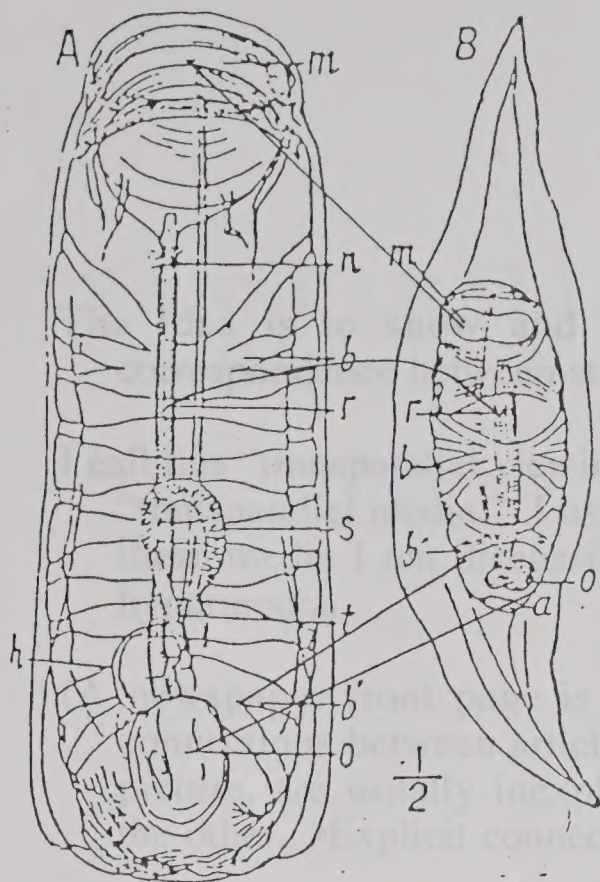
*Description of the Illustration:*  
*a* Right Ventricle of heart, with *b* Stump of Pulmonary Artery. *c* Left Ventricle, connecting with Aorta (*dd* Arch of Aorta, *ee* Descending Thoracic Aorta, *ff* Descending Abdominal Aorta), which shows the surface of the branching from it over the surface of the heart, and the stumps of the following arteries: *r* Right Carotid, *s* *s* Right and Left Subclavian, *t* Left Carotid, *u* Gastric, *v* Hepatic, *w* Splenic, *x* *x* Right and Left Renal, *y* *y* Right and Left Common Iliac, *z* *z* Right and Left Superior Mesenteric, *aa* Right and Left Inferior Mesenteric, *bb* Middle Sacral, *cc* Some of the Intercostal and Lumbar Arteries. *aa* Left Auricular Appendix. *r* Right Auricle and Appendix, receiving *ww* Superior Vena Cava and *xx* Inferior Vena Cava ascending from below, with stumps of Hepatic Veins.



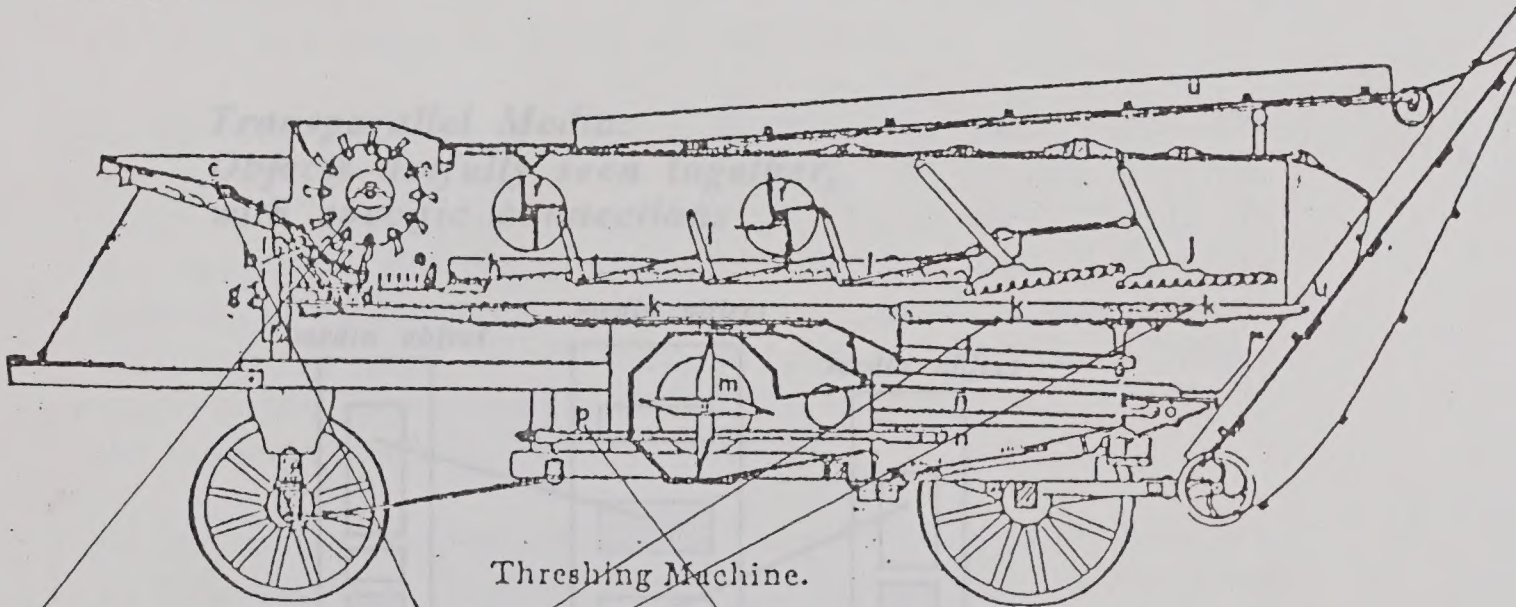








Salpa. A Solitary Individual.  
B Individual from a chain. m  
Branchial Aperture; a Atrial  
Aperture; b b' Muscular  
Bands; n Nerve Ganglion; r  
Dorsal Lamina; h Heart; o  
Nucleus; s Chain of young  
Zooids developing from the  
Stolon t.



Threshing Machine.

- a Cylinder; b Concave; c Feed Board; d Concave Adjuster;  
e Grates; f, f, Beaters; g Crank and Pitman driving Riddle  
(h, i, j) and Grain Pan (k, k'); u, u Straw Stacker, with apron  
folded for transportation; l Distributor; m Fan for blast;  
n, n Shoe with Sieves and Screen; o Tail Shoe; p Crank and  
Pitman driving Shoe; r Tailing Spout; t Grain Spout.

If we take appropriate inspiration from these examples, the class of software that it leads to is clear. This is not a new media concept, nor is it new in computers; but it is insufficiently understood or appreciated.







The idea is to show and see the detailed interrelations and correspondence between structures.

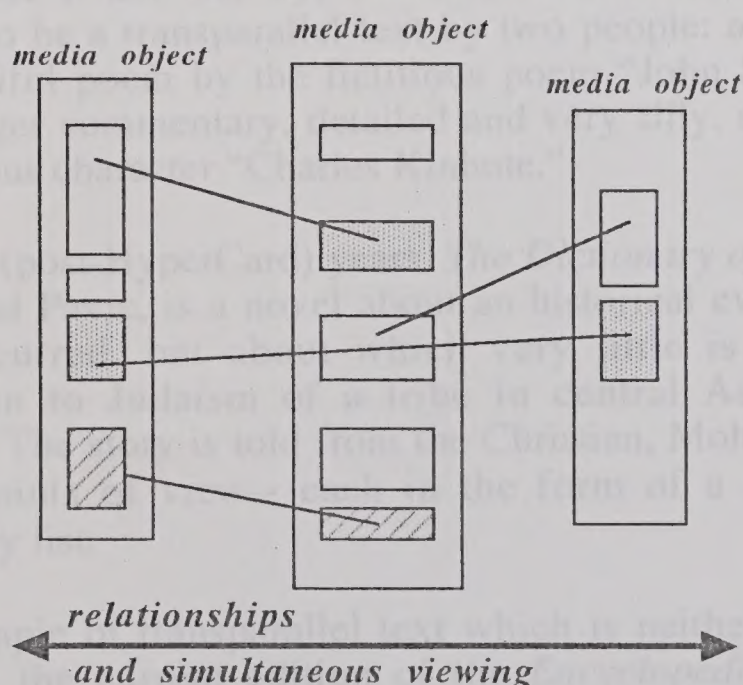
I call this “transparallel viewing,” and the corresponding media I call “transparallel media.” I use the term “transparallel” to distinguish these media I am discussing from other forms of hypertext and hypermedia.

(A newspaper front page is a PARALLEL medium, because the connections between articles, or between an article and a related picture, are usually inexplicit, with neither referring directly to the other. •Explicit connections make it transparallel.)

*Definitions.* By transparallel viewing, I mean seeing connections may be seen side by side between parts of objects

By transparallel media I mean media contents which are shown with detailed interconnections. (“Parallel” for viewing side by side; “trans-” because there is a specific connection to be seen between them.)

***Transparallel Media:***  
***Objects usefully seen together,***  
***with specific connections***







## TRANSPARALLEL TEXT MEDIA

There is a surprisingly large tradition of transparallel text, that is, text having corresponding parts. Most famous and perhaps the oldest is the Rosetta Stone, which showed the same text in several ancient written languages.

A number of key holy books have sections in common. Three of the world's main Bibles-- the Jewish Torah, the Islamic Koran and the Christian Old Testament share a number of parts, such as the Book of Genesis. Also sharing material with these official holy books are, certain books called the Apocrypha, which are ancient but not presently a part of any known religion. In the early years A.D., the Christian New Testament tells the same story a number of times with variations, which may also be regarded as transparallel. Around 300 A.D., the Hexapla, compiled by Origen, was an official comparison of six Bibles side by side. And the Jewish Talmud, which reached its present form approximately 900 A.D., is a system of interwoven texts centering on the Torah and commenting on each other.

Even the works of Shakespeare exist in transparallel form: *The Variorum Shakespeare* is an anthology of all the different versions of Shakespeare, especially for Shakespeare scholars. In the literature of this century, Vladimir Nabokov's novel *Pale Fire* pretends to be a transparallel text by two people: a long, serious and beautiful poem by the fictitious poem "John Shade," and a much longer commentary, detailed and very silly, supposedly by the fictitious character "Charles Kinbote."

In the recent (post-HyperCard) years, *The Dictionary of the Khazars*, by Milorad Pavic, is a novel about an historical event which we know occurred, but about which very little is known-- the conversion to Judaism of a tribe in central Asia called the Khazars. The story is told from the Christian, Mohammedan and Jewish points of view-- each in the form of a dictionary, or vocabulary list.

A final example of transparallel text which is neither fictional nor religious: the current edition of the *Encyclopedia Britannica*-- edited in the 1970s by Mortimer Adler-- consists of the same subjects covered at three different levels of detail and complexity.







These examples make it clear, I think, that transparallel text is a valid and important medium. It is hindered by the technology of printing and the inconvenience printed matter side by side--especially if the portions are of different lengths. (But these should be no hindrance in computerized versions.)



The system could show you things side by side, and also show exact connections among those things. An additional advantage is that with computer presentation, the number of things which may be shown together is unlimited, quite large, allowing for scrolling and zooming. The relationships may be seen side by side at any distance.

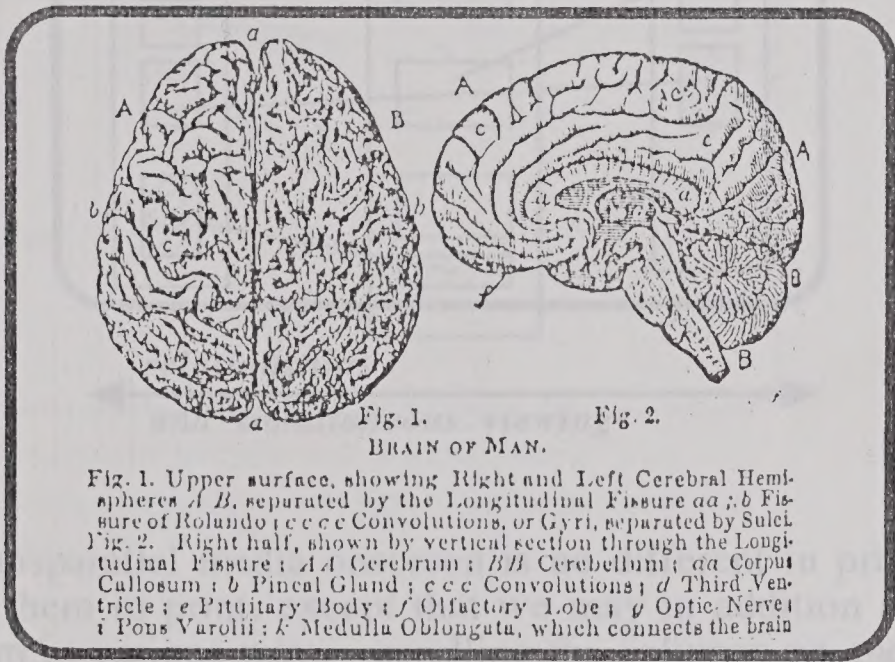
Transparallel screen media, then, will show the connections and correspondences between material and objects seen on the screen.





# COMPUTER TRANSPARALLEL MEDIA

Why, then, can't we use the same transparallel viewing techniques on the computer screen that have served us on paper? There is of course no reason that we could not see on the screen the same pictures that were in the dictionary:



The system could show you things side by side, and also show exact connections among those things. An additional advantage is that with computer presentation, the number of things which may be shown together is in principle quite large, allowing for scrolling and zooming. The relationships may be seen sideways to any distance.

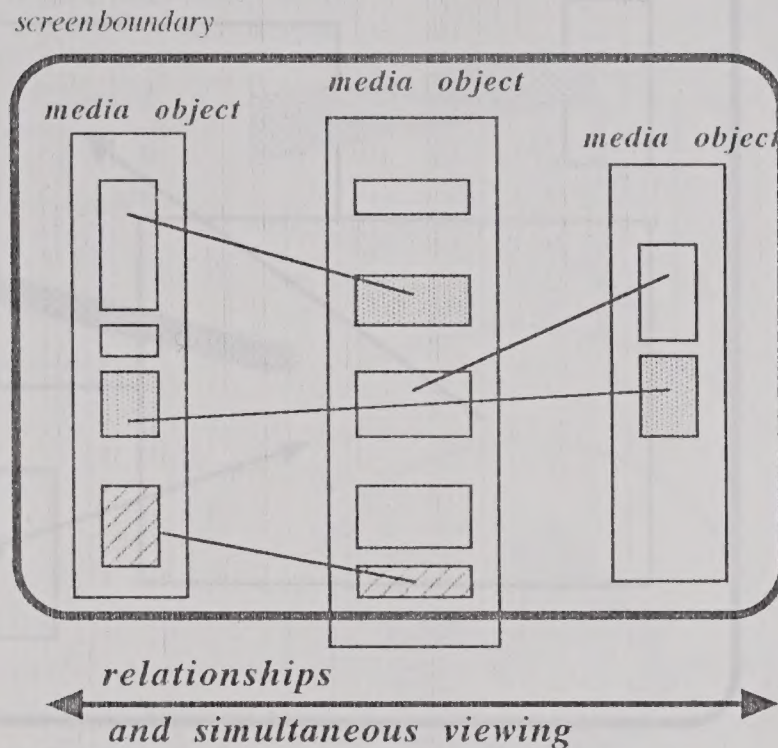
Transparallel screen media, then, will show the connections and correspondences between materials and objects seen on the screen.







*Transparallel Media onscreen:  
Objects usefully seen together,  
with specific connections*



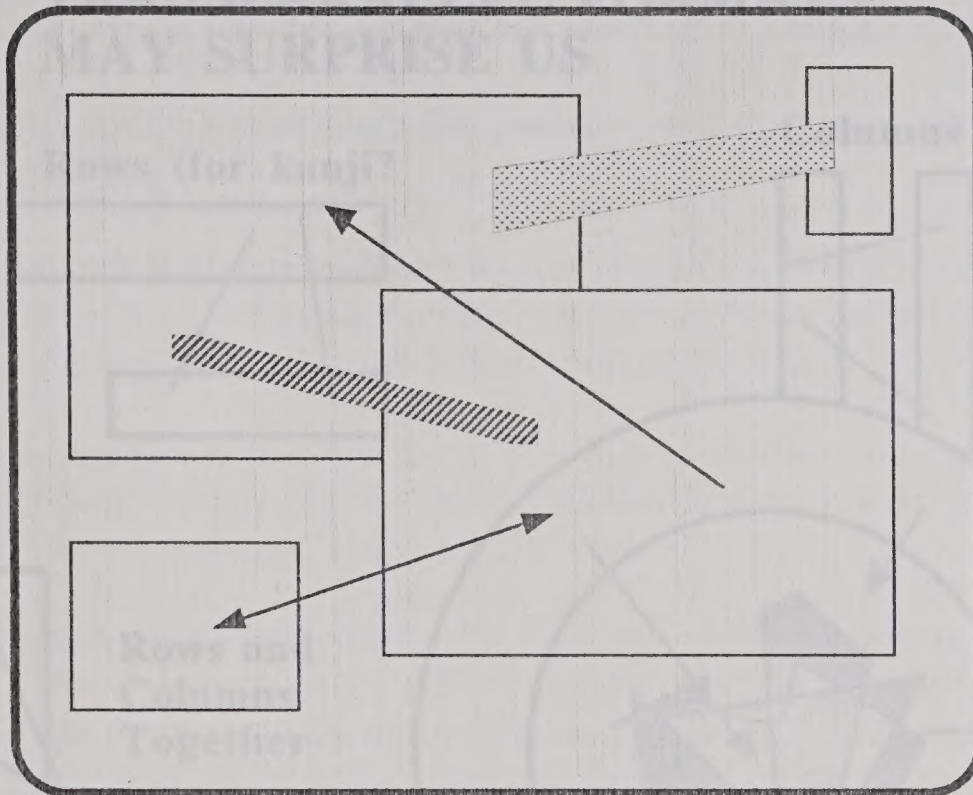
To show transparallel media onscreen is no different in principle from showing them in print, except that we may in addition animate and view them in various ways-- scrolling the columns up and down, for instance.

Note that the materials may not necessarily be "side by side" in vertical columns, but that appears to be a particularly useful visualization. A wide variety of other layouts is possible. The essential aspect of transparallel media is that specific connections may be seen between parts of things.





## TRANSPONING WINDOWS



Content connections are shown with explicit lines, bands and arrows among contents of screen windows (Nelson, 1969)

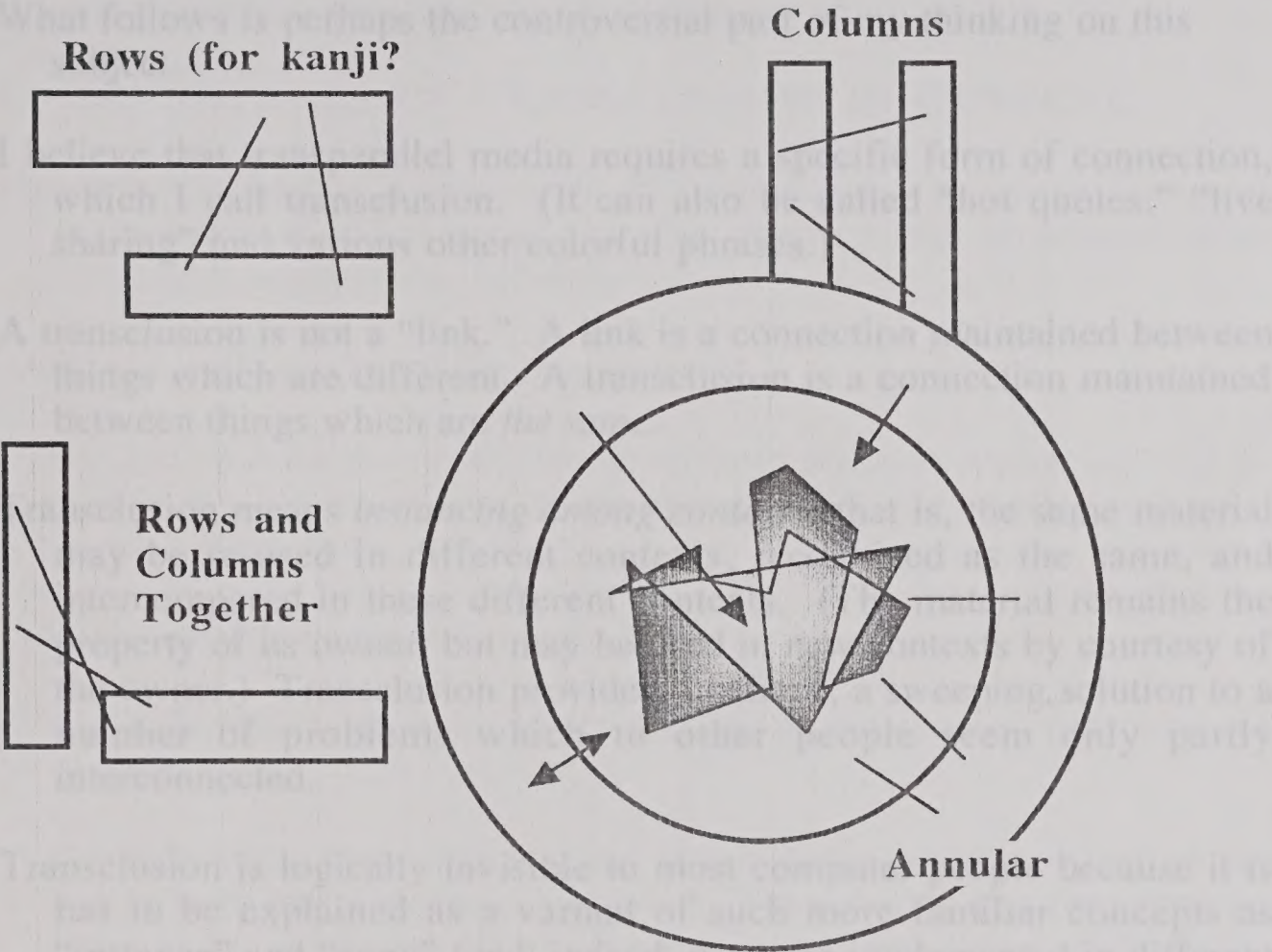
Although this report deals particularly with vertical columns, there will be many other transparallel formats.

Between transparallel screen presentation defines a new medium. It clearly leads to a new kind of side-by-side hypertext. And as a system for work, it offers the possibility of new kinds of overviews.





# TOMORROW'S TRANSPARALLEL FORMATS MAY SURPRISE US



Because transparallel screen presentation defines a new medium, it clearly leads to a new kind of side-by-side hypertext. And as a system for work, it offers the possibility of new kinds of overviews.





# TRANSLUSION FOR TRANSPARALLEL MEDIA

What follows is perhaps the controversial part of my thinking on this subject.

I believe that transparallel media requires a specific form of connection, which I call translusion. (It can also be called "hot quotes," "live sharing" and various other colorful phrases.)

A translusion is not a "link." A link is a connection maintained between things which are different. A translusion is a connection maintained between things which are *the same*.

Translusion means *instancing among contexts*: that is, the same material may be re-used in different contexts, recognized as the same, and intercompared in these different contexts. (The material remains the property of its owner, but may be used in new contexts by courtesy of the owner.) Translusion provides, I believe, a sweeping solution to a number of problems which to other people seem only partly interconnected.

Translusion is logically invisible to most computer people because it is has to be explained as a variant of such more familiar concepts as "instance" and "copy" (and, indeed, it can be implemented in different ways using instances and copying.) The logic and power of translusion take some explaining.

\*/\*The logic of translusion is separate from the implementation. It may be implemented in various ways, provided that the identity of the materials is maintained.

Translusion does not mean conventional copy or transfer of bytes, whereby the bytes lose their identity; it means *identified re-use allowing intercomparison and retaining ownership*. This requires keeping all the re-used materials separate and identified as to their origins, so that each item has a pathway back to its original. These pathways provide bridges of context-- between the new and original context-- for viewing and other purposes.







In the fullest form of transclusion, there is full transvisibility between all instances: all instances of the same material are linked sideways, as it were, so they may be viewed together sideways, as it were, in a transparallel display showing all the uses of any item.

## 1. ADVANTAGES OF TRANSLUSION IN GENERAL

Transclusion means re-use with understanding. It means the easy repurposing of all materials in new contexts and structures, with intercomparisons among all these settings.

And there will be a key difference from all

But here will be the main differences from previous media: we will be able to move sideways out of one object or title into another, and to re-use the material.

Transclusion allows the following:

Multiple uses of an item, under control; eg versions of text, versions of programs.

Cross listing, cross-indexing and cross-filing: the one virtual object can be in as many places as you want. Transclusive structure allows a thing to be in several places at once, and allows those places to be connected.

Identifiable re-use of the same material in a number of different contexts.

The ability to compare the re-uses of the material in their different contexts.

A potential cleanup of the copyright problem.\*

\*/\*In the transclusive copyright system, a reader would essentially purchase an item, or a portion, thus retaining both that piece and the pathway to its original context, for possible later use. (This has recently been updated as the doctrine of “transcopyright” (4).





## 2. IS TRANSLUSION NECESSARY FOR TRANSPARALLEL SCREEN MEDIA?

It is my strong belief that translusion is necessary for screen transparallel media.

There are two possibilities: transparallel media without translusion, where we can only connect things, but not re-use them. This is similar to HyperCard and World Wide Web. The connections allow certain types of connections, but not the re-use of material in new contexts.

Thus transparallel media without translusion would allow unwitting materials to be linked and shown side by side, but no more.

Transparallel media with full translusion completely change the game, allowing everything to become clip art, and allowing new constructions to use old ones without restriction.

There is no technical argument for this. It is a media argument. I believe that media which add this re-use feature will have extraordinary new power for the human mind and society. That is today only a credo and no more. I hope that soon it will be a fact demonstrable to all.

## LINK AND TRANSLUSION ARE COMPLEMENTARY

While it is not part of the definition, in most of the xanalogical systems I have worked on, these relations of link and translusion have been *applicative*, and can therefore be applied to bodies of data





# THE XANALOGICAL FAMILY OF MEDIA

I propose a family of media and the software to enact it.

The xanalogical family of media is a family of designs that I have been advocating and working on for a considerable length of time. People are familiar with certain aspects of the ideas, but not with others.

The xanalogical family of media ranges from the simplest coarse-grained (zipper lists (1)) to the most complex (very fine-grain distributed network publishing (2,3)) with the same basic structure.

Xanalogical media are defined as follows: media (environments, contents and works within them) which are connected in either of two ways: transclusion and link.

A variety of xanalogical media are possible, but they are an order of magnitude more complex than today's simple hypertext and hypermedia (such as HyperCard, MacroMind Director, World Wide Web).

Such media can range from coarse-grained media, such as zipper lists (1), to the extremely fine-grained media offered by the fine-grained hypermedia servers still under development at XOC, Inc., described in Literary Machines (2) and the XOC DOC (3).

The supporting software can range from relatively small programs (such as the HyperLab Zip Editor) to large-scale server networks (2, 3).

## LINK AND TRANSCLUSION ARE COMPLEMENTARY

While it is not part of the definition, in most of the xanalogical systems I have worked on, these relations of link and transclusion have been *applicative*, and can therefore be applied to bodies of data





(especially text and pictures) without changing them or embedding codes. This is a particularly good feature for legacy and unchangeable applications, especially in the text area. (4?)

## **GENERAL, EXPANDABLE AND ORDERLY**

The xanalogical family of designs has been designed to have the greatest generality in directions which I think are important, representing connected electronic documents in a way that is intended to be simple and clarifying.

I believe this is an orderly, expandable and understandable system, a system that people of all ages and levels of education can learn to use easily, like books-- although of course some electronic documents may be difficult to read, like many books.

## **MANY DIFFERENT POSSIBLE IMPLEMENTATIONS**

Xanalogical media may be implemented in many different ways. The type of implementation is indifferent, provided that the logic is followed in the correct manner.

## **MANY DIFFERENT POSSIBLE INTERFACES**

Xanalogical media center on the structure of the data-- the contents and connections. The particulars of any interface are a matter of interest but not central to the concept.

## **APPLICATIVE STRUCTURE**

In the design of xanalogical media, it has always been apparent that structure would have to be separate from bytes (or other content elements), in order to facilitate re-use and quotation. Organizing





structures, links or larger structures (if any), can be applied to any data. These are “applicative,” meaning that they take effect from outside without getting in the way of other uses of the data.

The classic 1965 zip editor design was implicitly an applicative design, since the item was logically distinct from the link; a link could be added and removed independently of the revision of the item.

(This system of managing structure on an applicative basis meant avoiding embedded codes, like those of such later developments as link anchors, HTML or Microsoft Word.)

Thus links would be kept in the equivalent of a separate store from primary data (text and other contents). (There are no special mechanisms for special link types (EXCEPT TRANSLUSION, which is NOT A LINK).\*

\*Note that by this reasoning transclusions should in principle also be kept separate from the data--except that they are already merely implicit in the structure. There is no “transclusion data;” transclusions must be found.

Other organizing structures, if any, would likewise be kept separate.\*

\*In the Xanadu 88.1 design, it was intended that links could be built and combined into higher applicative structures. In the present Xanadu designs, other applicative organizational forms and templates are being considered.





## ZIPS, THE SIMPLEST CASE

Zips (zipper lists) are an abstract simplification of xanalogical structure. Zip structure is are fairly easy to implement, but allows experimentation with central problems of visualization, understanding, manipulation and transclusion.

Zips are a large-grain, chunk transparallel medium. Thus I see zips as the simplest example of tomorrow's basic information structure. The design of zips, first presented in 1965 (1), is still valid, as demonstrated by our work at HyperLab. (Note, however, that the 1965 design is the reference design for purposes of discussion.)

### WHAT ARE ZIPS? (informal)

Zips are a special kind of information, and visualization for it, which can be more or less used for everything if you want to, much like a word processor or spreadsheet. (Note that zips were defined before either word processing or spreadsheet existed.) Because of the relations automatically maintained by the system, zip structure will help the user keep track of things more easily than word processing or spreadsheet.

Informally stated, zips are side-by-side columns with criss-cross connections. Or columns of paragraphs which may be viewed side by side with visible connections between the columns.

Zip editing is working on their connected sequences of items, either reworking the pieces themselves or the overall structure. Zip editing is basically viewing and rearranging lists, whose transparallel connections are maintained automatically.

The user creates zips, creates items, creates connections, and chooses what zips to see at a given time. The system automatically shows the connections between whatever zips are seen.

### WHY DO YOU DO IT?

The purpose of zip editing is to see, work with, understand and control complex relations among related materials, especially text, or





## ZIP

|      |
|------|
| item |
| item |
| item |
| item |
| item |
| item |
| item |

## ZIP

|      |
|------|
| item |
| item |
| item |
| item |
| item |

## ZIP

...

Zips may be used as an under-structure for cross-filing, intercomparison, showing the evolution of versions, for commentary on corresponding parts of two objects, and more (see later section, “Defining Some Zip Functionalities.”)

## WHAT IS ZIP EDITING?

“Whatever you can do with a word processor, but better.”

Zip editing is working on such connected sequences of items, either reworking the pieces themselves or the overall structure. Zip editing is basically viewing and rearranging lists, whose transparallel connections are maintained automatically.

The user creates zips, creates items, creates connections, and chooses what zips to see at a given time. The system automatically shows the connections between whatever zips are seen.

## WHY DO YOU DO IT?

The purpose of zip editing is to see, work with, understand and control complex relations among related materials, especially text, or





interrelated things which the text represents.

Using zips and items as the building blocks, it is possible to create large complexes of data for a personal working environment. This is best expressed in the illustration from the 1965 paper:

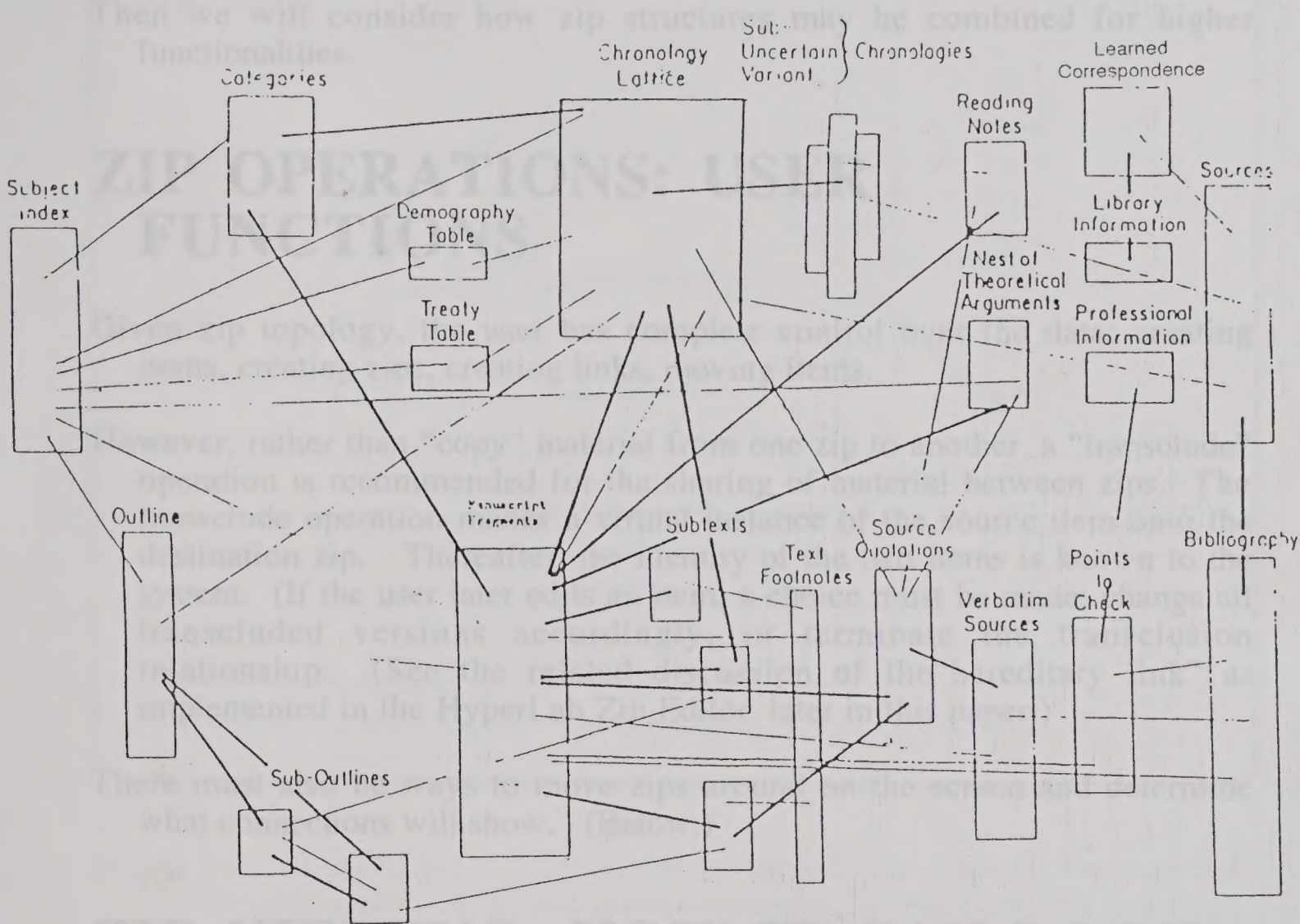


FIGURE 4- ELF's capacity for total filing: hypothetical use by historian. (A thin line indicates the presence of links; a heavy line indicates that some linked entries are identical.)





## SPECIFICS OF ZIP SYSTEMS

In what follows we will discuss the exact definition of zip structures, their appearance, and basic user functions.

Then we will consider how zip structures may be combined for higher functionalities.

### ZIP OPERATIONS: USER FUNCTIONS

Given zip topology, the user has complete control over the data: creating items, creating zips, creating links, moving items.

However, rather than “copy” material from one zip to another, a “transclude” operation is recommended for the sharing of material between zips. The transclude operation makes a virtual instance of the source item onto the destination zip. Thereafter, the identity of the two items is known to the system. (If the user later edits an item, a choice must be made: change all transcluded versions accordingly, or terminate the transclusion relationship. (See the related discussion of the “hereditary link” as implemented in the HyperLab Zip Editor, later in this paper.)

There must also be ways to move zips around on the screen and determine what connections will show. (Below.)

### ZIP VIEWING: HOW IT CAN LOOK

Zip viewing is intended to allow you to see connected materials side by side. Items may be connected by visible lines.

Since the subject matter of a zip editor is items, zips, links and transclusions, those are what needs to be shown in the interface. There need be no tricks.

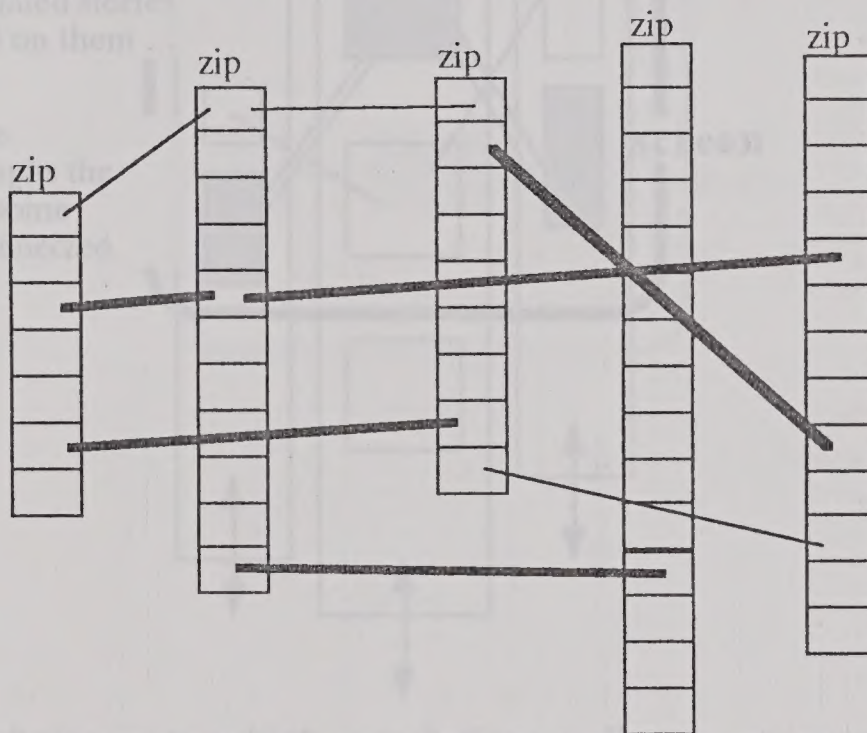




## Zipper Lists: The Interface Can Look Like the Data Structure

Connections and contexts are simultaneously visible.

*(Possible  
overview)*



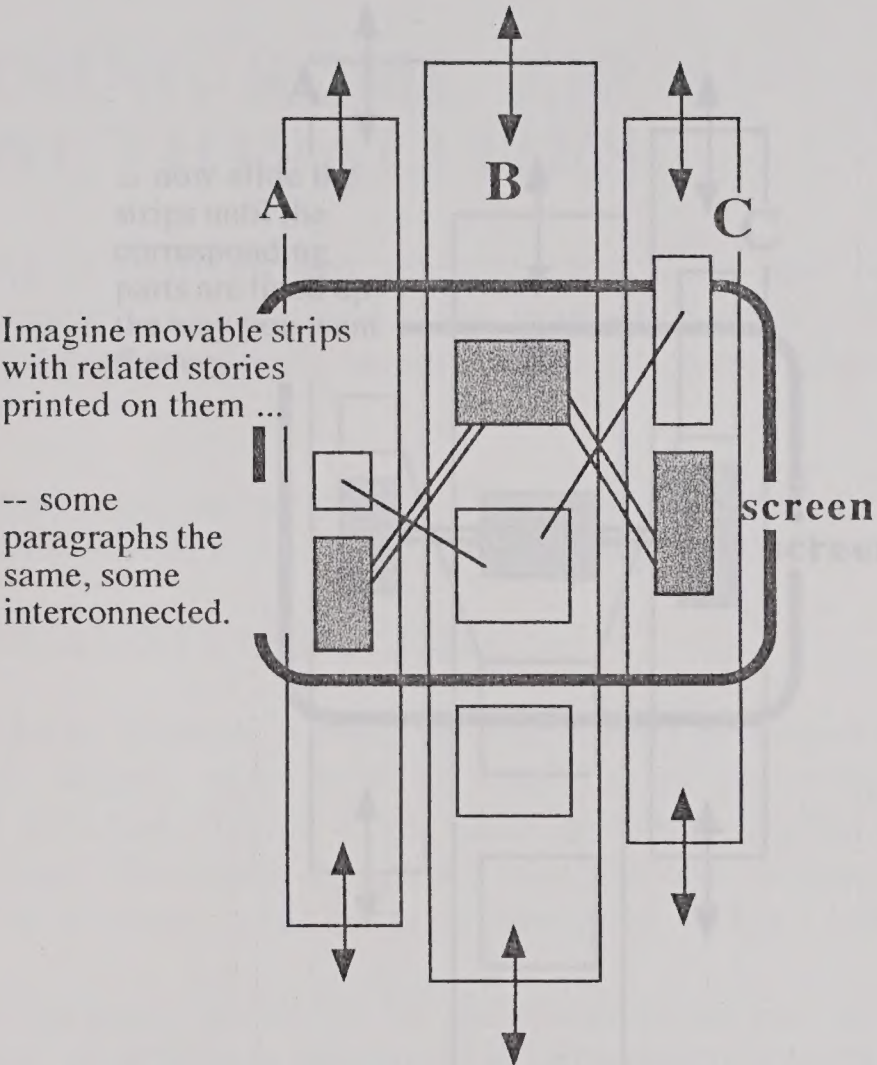
— Heavy lines: same paragraphs (transclusions)  
— Light lines: other connections (links)

Zipper lists are intended to let you line up the relevant parts of the material. Besides being able to see the interconnections in overview, the user may maneuver them into different views and magnifications, to study the interconnections and contexts close together.

Since parts of zips may be connected side by side in ways that would be inconsistent in a paper world, the user gets to decide from moment to moment what parts to view together.





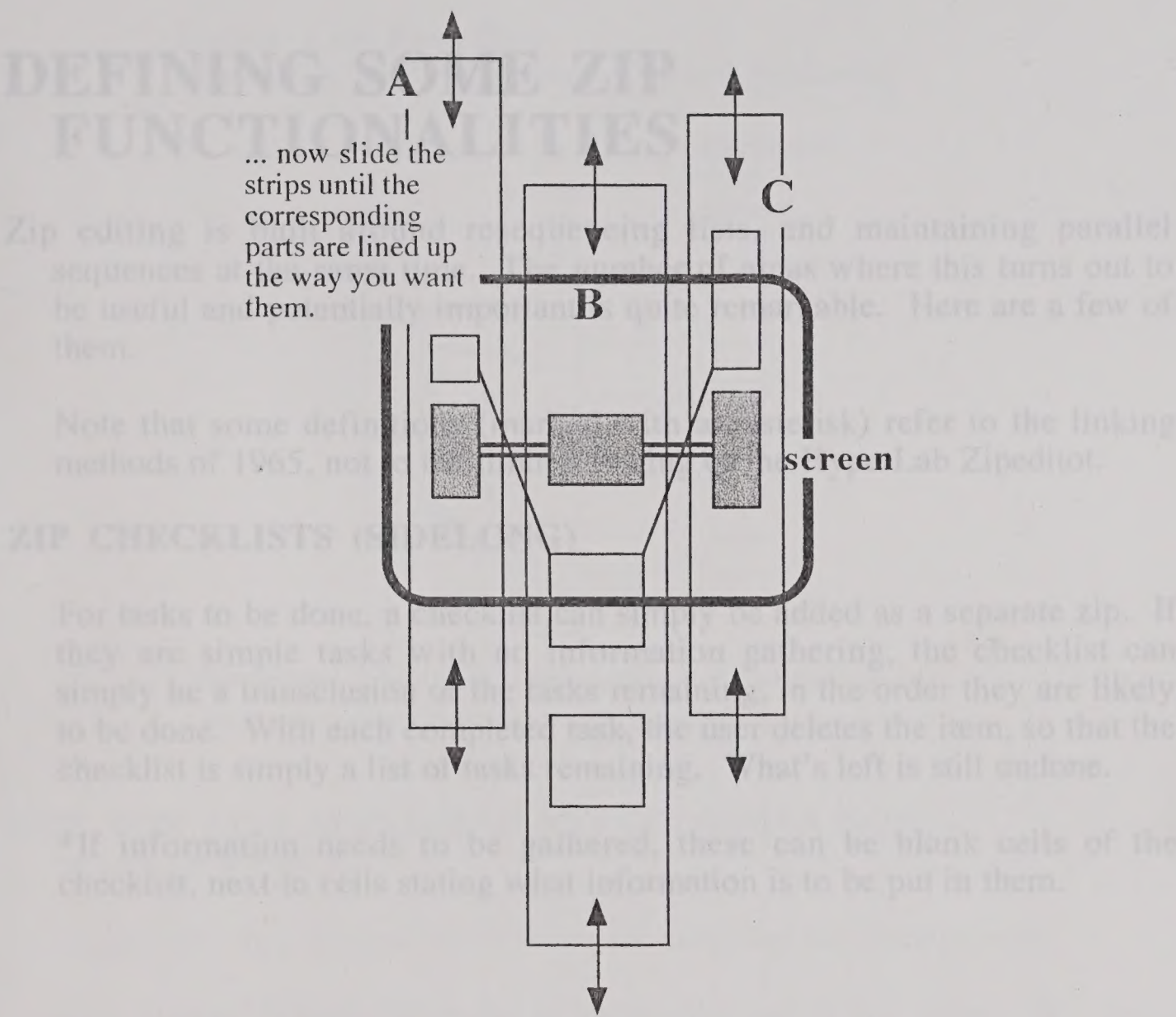


The user has a choice as to which connections to line up at any given moment; some possible viewing modes would allow many connections to be lined up simultaneously, with some loss of overview and less visual coherence for the individual zips.

Thus it is important to be able to move the views in rapid maneuver.







Thus it is important to be able to move the views in rapid maneuver.





## DEFINING SOME ZIP FUNCTIONALITIES

Zip editing is built around resequencing lists, and maintaining parallel sequences at the same time. The number of areas where this turns out to be useful and potentially important is quite remarkable. Here are a few of them.

Note that some definitions (marked with an asterisk) refer to the linking methods of 1965, not to the limited linking of the HyperLab Zippeditor.

### ZIP CHECKLISTS (SIDELONG)

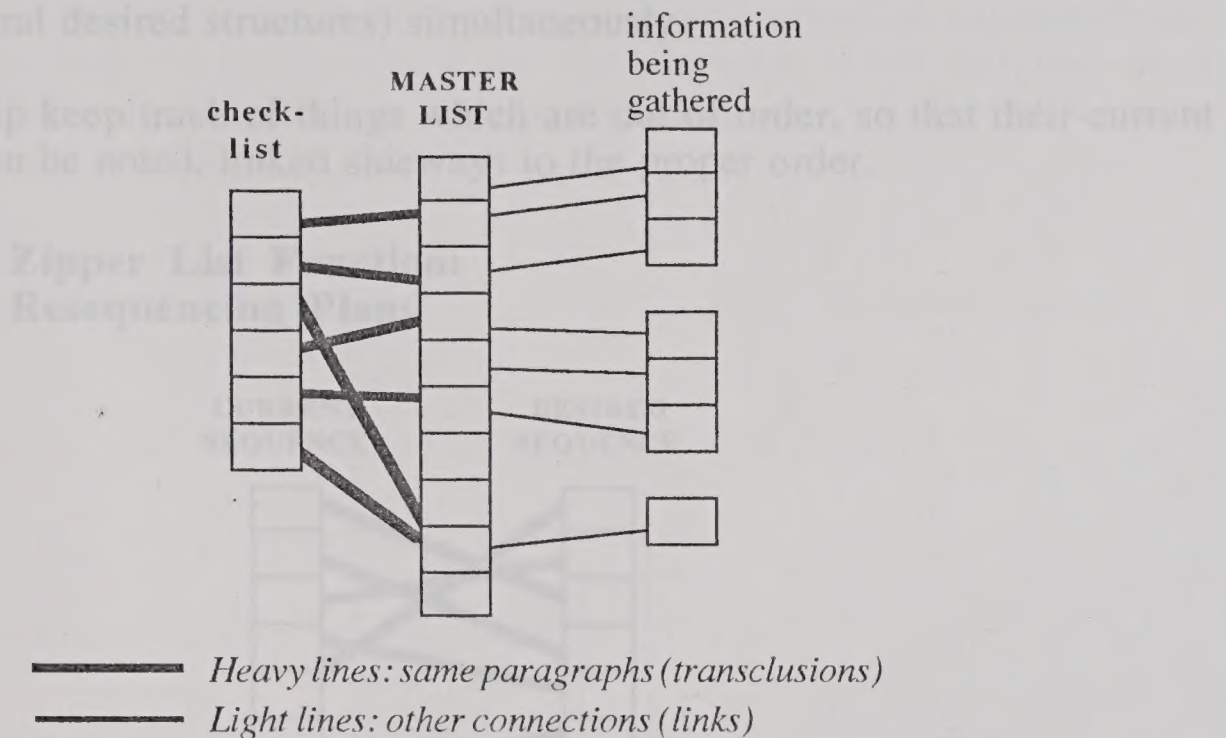
For tasks to be done, a checklist can simply be added as a separate zip. If they are simple tasks with no information gathering, the checklist can simply be a transclusion of the tasks remaining, in the order they are likely to be done. With each completed task, the user deletes the item, so that the checklist is simply a list of tasks remaining. What's left is still undone.

\*If information needs to be gathered, these can be blank cells of the checklist, next to cells stating what information is to be put in them.





## Zipper List Function: Checklists and “Clipboard” data gathering



## ZIPS FOR LIST MANAGEMENT, COORDINATION, MAINTENANCE

Zipper lists may be used as facilities for everyday list management.

Our human accomplishment is generally based on lists. We must constantly make and revise lists for every purpose of getting things done.

Transferring from a master list to sublists is a constant nuisance. With zips, it is possible quickly to transclude the appropriate material from one zip to another.

## \*ZIPS FOR PAIRWISE ASSOCIATION, N-WISE ASSOCIATION

Zips are particularly suited to keeping track of “Who’s paired up with whom”-- what tutor has been assigned to what student, or a breeding schedule for prize cattle, or who is supposed to do what task.

This gives an overview of what things are associated with what others, who is handling a job, etc.



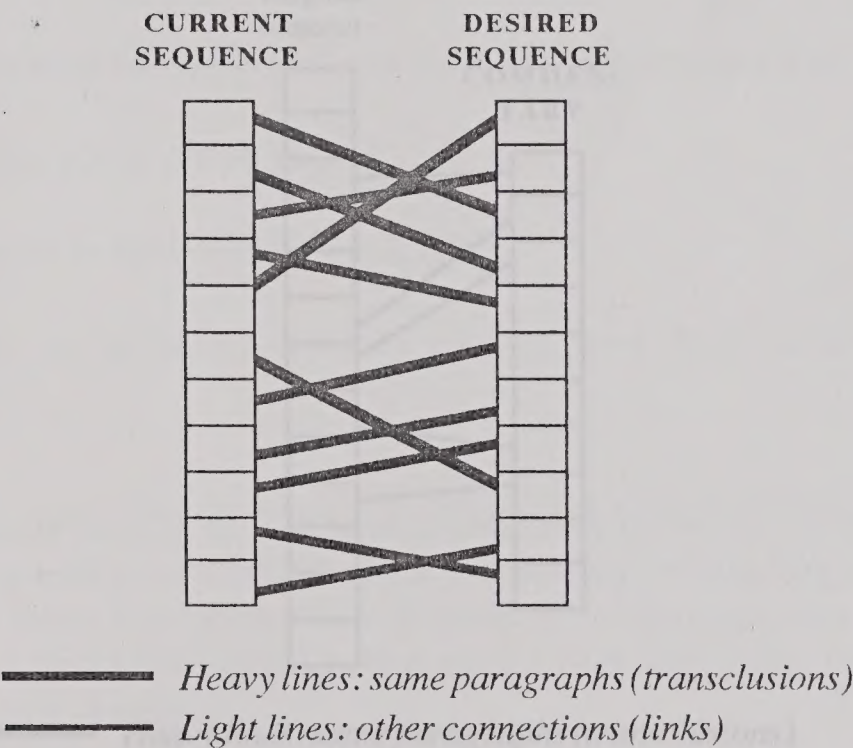


# ZIPS FOR RE-SEQUENCING AND REORGANIZING

Zip structure can show both the current structure and the desired structure (or several desired structures) simultaneously.

Zips help keep track of things which are out of order, so that their current order can be noted, linked sideways to the proper order.

## Zipper List Function: Resequencing Plans



This helps with overviews of rearrangement, or of the reassembly of many pieces into some new order.

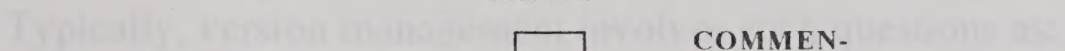
A simple example is the resequencing of audiotape, in a planned edit. The current sequence is one zip, the desired sequence is the other zip.





Zips allow the user to make commentaries or annotations on any document.

## COMMENTARIES



Where did this piece come from?

|                                |      |
|--------------------------------|------|
| What happened to that section? | XXXX |
|--------------------------------|------|

How was this in an earlier version? (It may have been longer)

Many technical people do not understand the problem. Writers and

Traditionally have not been. ☐ attitude of writers and lawyers about





## ZIPS FOR VERSIONING

Zip structure can show clearly the successive changes among versions of a document, particularly of rearrangements. We can view them side by side, in marching transparallel versions.

Fully-transcluded paragraphs are shown directly. Partly-transcluded or rewritten material becomes a link.

If this does not show the changes accurately enough, the user can simply declare new versions more often.

Typically, version management involves such questions as:

Where did this piece come from?

What happened to that section on XXX?

How was this in an earlier version? (it may have been better)

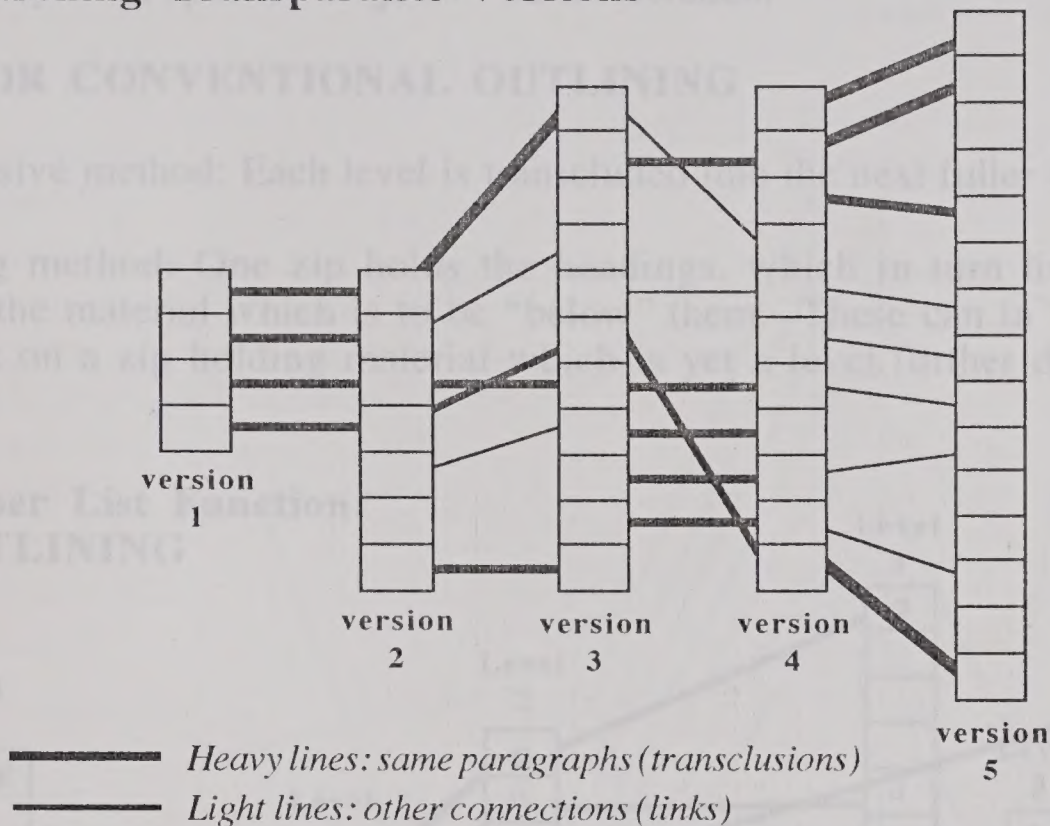
Many technical people do not understand this problem. Writers and lawyers are greatly concerned about versioning, while engineers traditionally have not been. The attitude of writers and lawyers about versioning is best expressed in an e-mail I received from Jim Bryce Clark, a lawyer in Los Angeles:

- > In my field (law), twenty or thirty successive versions of a document are
- > not uncommon. The idea of a "final" document, the finality of which
- > justifies throwing out all the historical drafts and previous work, is fast
- > becoming obsolete.





### Zipper List Function: Marching Transparallel Versions



## ZIPS FOR E-MAIL THREADING-- an example of COMPLEX STRUCTURE

Zips can be used to represent complex structures of association of sequential objects. E-mail threading is a good example.

In conventional "threaded e-mail readers," an intertwined narrative sequence of e-mails is filed according to the titles of specific e-mails. Anyone who uses e-mail, however, knows how little the titles correspond to the actual subjects discussed; and the problem that one e-mail message may be relevant to a number of separate topics. In the current system (or lack of one), there are basically two ways to handle the problem: 1) file the same message under as many headings as necessary. 2) Quote it into an ongoing narrative of the topic (too much trouble).





## ZIPS FOR CONVENTIONAL OUTLINING

\*Linking method: One zip holds the headings, which in turn link to a zip holding the material which is to be “below” them. These can in turn link to elements on a zip holding material which is yet a level further down in the outline.

**Zipper List Function:  
OUTLINING**

**Level 0**

**TITLE**

Commentary

**Level 1**

**a**

**b**

**c**

**Level 2**

**a**

d

e

**b**

f

**c**

g

h

**Level 3**

**a**

d

**c**

f

g

h

**Level 3**

**b**

**c**

**Level 3**

**h**

**(Method of Transclusion)**

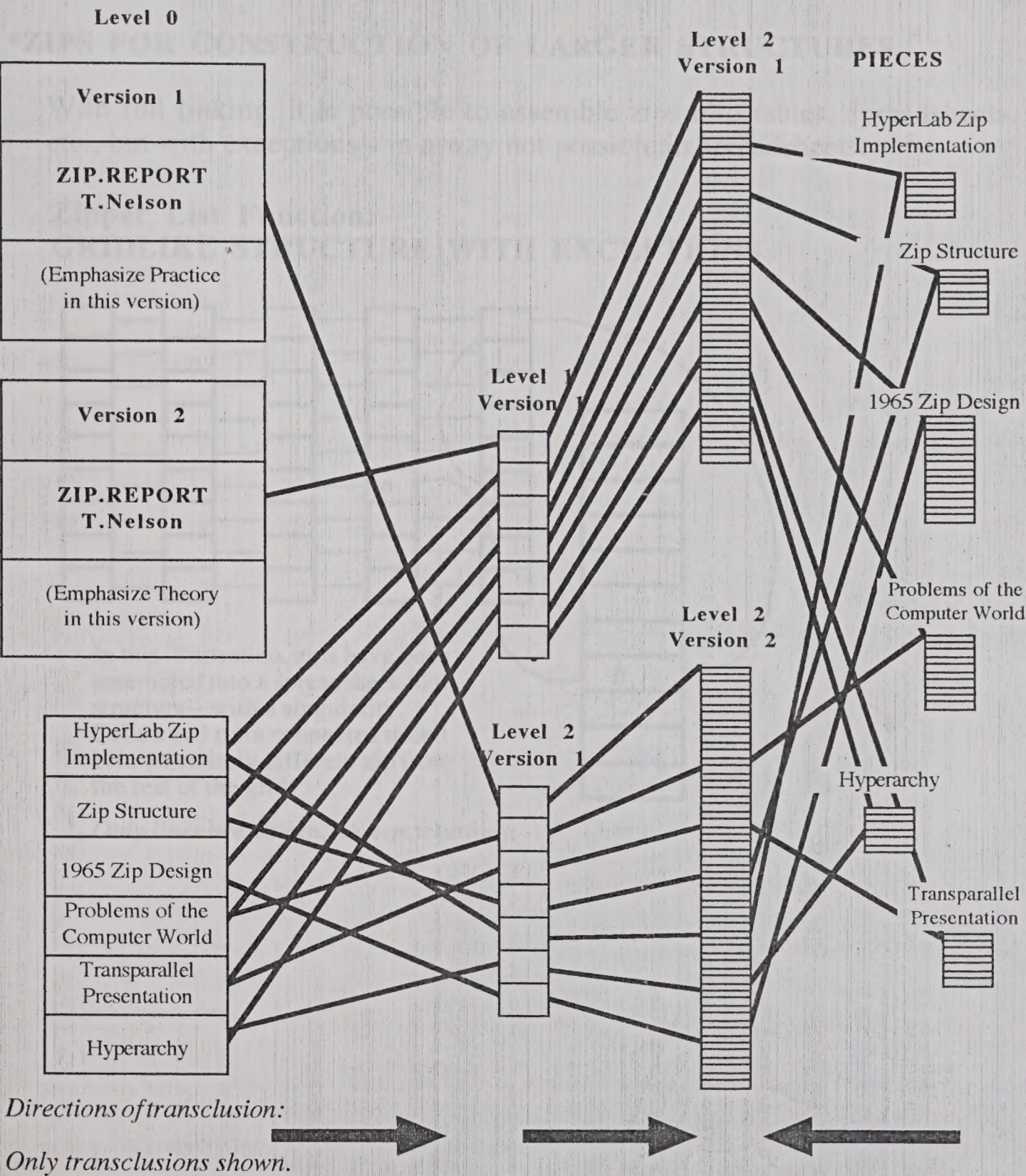
In hyperarchical outline, more than one hierarchy may be applied to a given body of material at the same time. This can work identically to conventional outlining, except that more than one outline may be seen to be simultaneously operative.





# Zipper List Function: HYPERARCHICAL OUTLINING

(Method of  
Transclusion  
shown)







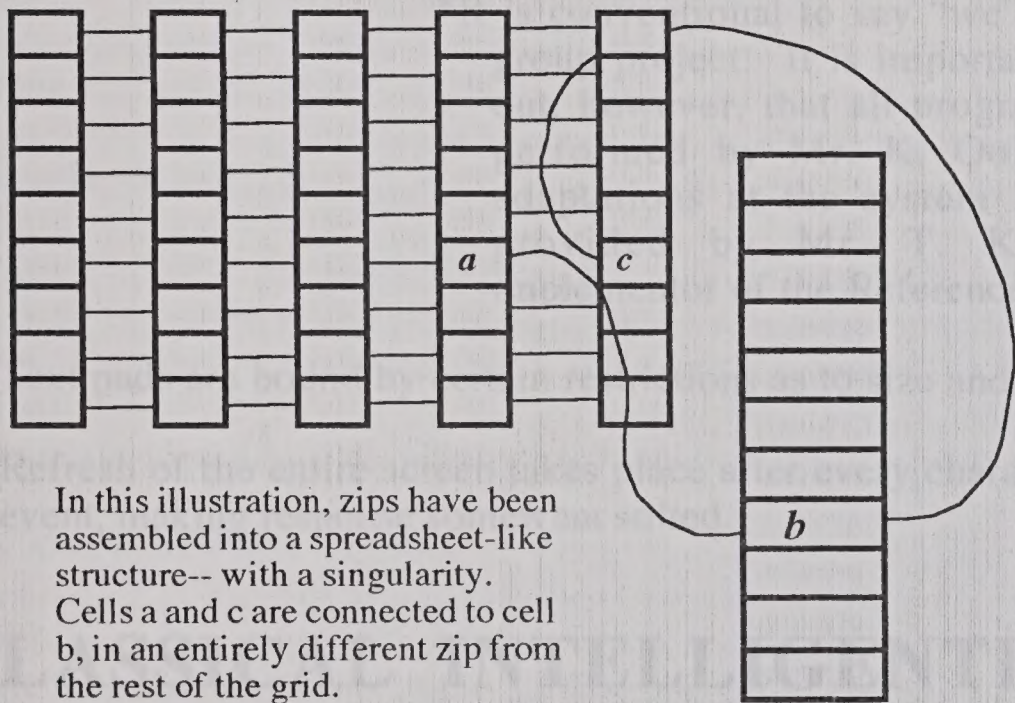


This can provide an overview of version differences between the different organizational plans under consideration.

**\*ZIPS FOR CONSTRUCTION OF LARGER STRUCTURES**

With full linking, it is possible to assemble zips into tables, spreadsheets, etc., but with exceptions-- in a way not possible in spreadsheets.

**Zipper List Function:  
GRIDLIKE STRUCTURE WITH EXCEPTIONS**



In this illustration, zips have been assembled into a spreadsheet-like structure-- with a singularity. Cells a and c are connected to cell b, in an entirely different zip from the rest of the grid.

*Only links are shown, no transclusions.*







## THE HYPERLAB IMPLEMENTATION OF ZIP EDITING

Because of the historical timing of the project, we\* implemented the Zip Editor using the Reference Version of IntelligentPad. The Reference Version of IntelligentPad, implemented using the InterViews window management package, provided an excellent baseline system for study and experimentation in the 1994-5 time-frame. However, because of this windowing package developed elsewhere, the IntelligentPad Reference Version had certain overhead penalties and presented certain design challenges:

\*It is conventional to say "we" with regards to a group project. It is important for me to point out, however, that all programming work was performed by Mr. K. Ookubo. Particular adaptations at the system level were kindly provided by Mr. T. Kainuma, system implementor of the Reference Version.

Text pads are bound by certain restrictions as to size and editability.

Refresh of the entire screen takes place after every character input and user event, making response somewhat stilted.

## CLASSICAL INTELLIGENTPAD

IntelligentPad has a remarkable relationship between interface and contents. Typically each pad is visible, and represents a specific function. The user deals with individual pads which both carry out the function and may be incorporated in the interface. Thus, in theory, the interface and the program are substantially the same, and the interface is the same as of the pad structure.

It should be noted, however, that this enacts the geometrical constraints of layers of paper, successively smaller, and so exceptions to the paper model - such as the Zip Editor-- need to be creatively adapted to this environment.







# ADAPTATIONS TO THE INTELLIGENTPAD ENVIRONMENT

Working within these constraints, however, provided valuable experience as to the relative importance of various aspects of such systems.

Conventional windowing systems, such as OpenWindows and Tcl, have provisions for line drawing. The emphasis of IntelligentPad is different, and based on a hierarchical pasting model, so that a special line pad had to be implemented at the lower level, transparent except for the displayed line.

Data management functions, traditionally separated on the Xanadu project from issues of interface, continued to be so separated here-- not for any doctrinal or design reason, but because of the two separate implementation levels, IP and Tcl.

Because of IP's programming hooks to Tcl, the data management features could be entirely implemented in Tcl, a scripting language with close ties to C. This meant that effectively there were two levels of programming: the IntelligentPad (IP) layer, concerned principally with interface, and the data level, essentially implemented in Tcl.

This also meant that modules of the data management code could successively changed at the Tcl level. An early version storing the items one way was replaced by a module storing them another way, yielding much better performance on transclusion lookup and other aspects.







# ADJUSTMENTS TO THE 1965 DESIGN 1: THE HEREDITARY LINK

The 1965 design (Appendix 0) had a multi-linking facility that would allow any item to link to any number of other items, as long as each item was on a different zip, with untyped links. (This was implicitly an applicative design, since the item was logically distinct from the links, which could be added and removed independently of the revision of the item.)

As a practical adjustment in the current project, however, that would have added one level of indirection and introduced further complexity to the system, which it was important to avoid. (We also wanted to experiment with heredity creation of text materials.) Therefore a more limited linking facility was added, where each pad could link hereditarily to its parent pads and its children.

The way it works is this: when an item is copied, it maintains a strong connection (transclusion) to its original and any other exact copies of the item. However, when such a copied item is modified, this connection to the parent becomes a hereditary link (which it maintains to all instances of the parent).

This can be used like the more general links of the 1965 design, but not as freely.

In order to do this, you make a copy of the item, then revise it, replacing it with any other text. You now have two distinct pads connected by a link, which may cross between zips. This can indeed be done more than once. But this method results in links which have a variety of detailed differences from links in the 1965 design, which we need not go into here.







## **ADJUSTMENTS TO THE 1965 DESIGN 2: SHOWING THE CONNECTIONS**

In the 1965 design, it was not made clear how the visible display would show the connection-lines among items. If a connection-line was shown between an item and all its link-mates and transclusion-mates, the number of lines on the screen could easily explode into a vast tangle of connection-lines.

The simple solution chosen in the HyperLab Zip Editor was to show only the transclusion-lines to the nearest horizontal transclusion-mate.

## **USER REACTIONS**

Because of the vicissitudes of the implementation, there have been few users of the system. Several of those who have seen it demonstrated, however, expressed surprise at the concept and said they would like to have such a thing when it becomes a real product, rather than a prototype.

However, this is just an impressionistic reaction, since the number of people who have seen it demonstrated has been about thirty to this date, and the number that volunteered as to how they would like such a system was approximately three or four people.

We look forward to showing this to more people for a broader sense of public reaction.





## ZIP-BASED LITERARY FORMS

When zip editors become popular, they will at first be used only as tools. But literary tools engender new literary forms. Beginning as facilities, zips will become media.

New literary forms will be possible using only zips, especially chunk transparallel collateral hypertext (writings linked in parallel columns which do not have the same sequence). I expect that zips may be used for the actual creation and use of transparallel hypertext-- as *zip documents* -- the simplest transparallel media for the creation of works and presentations.

A Zip Document will thus consist of a group of connected zips (or zipplex) holding material on the same subject, or on connected subjects.

The zip document consists of a number of zips with shared or connected cells; each is a coherent sequential subset of the subject-- an article or story.

Because as yet no one has had the opportunity to create documents using this structure, not know what types of literary structures and effects will be possible or useful.

However, I have been personally designing a zip-based hypertext, and one of the principal features of that design is what we may call a *transclusion tunnel*, or possibly *transcrostic*.

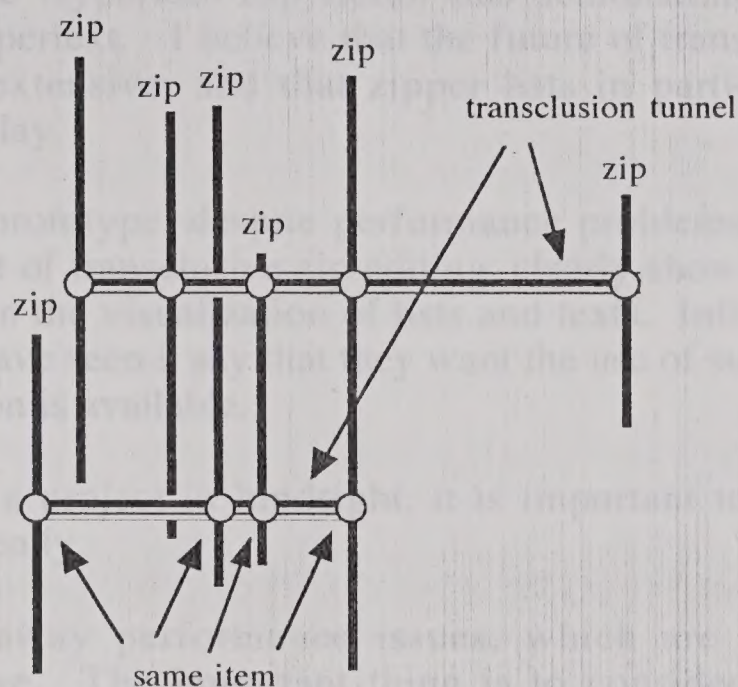






## ZIP DOCUMENTS-- for making complex subjects, and their connections, understandable.

Connections and their contexts are simultaneously visible.



A transclusion tunnel is the repeated transclusion of the same item across a large number of zips, to draw attention to it and emphasize its importance. This is a new way of giving overall form to hypertext.







# CONCLUSION

Many people wonder about hypertext, and the difficulty of visualizing it as a whole. But the HyperLab Zip Editor has demonstrated a new type of overview for hypertext. I believe that the future of transparallel computer media will be extensive, and that zipper lists in particular will have a healthy role to play.

Our experimental prototype, despite performance problems, has achieved a proof-of-concept of transclusive zip editing, clearly showing the usefulness of transclusion in the visualization of lists and texts. Informally, a number of people who have seen it say that they want the use of such a facility when a polished version is available.

When considering a project in hindsight, it is important to review what we would do differently.

We will assume away performance issues, which are “not there” in a substantive sense. The important thing is to consider the logic of the system. In the next implementation, I would like to see a fuller linking system, on an applicative basis, to allow work with the complete 1965 design. A number of the key topologies in that design are out of reach of the current version. It would also be good to have a test bench for other zip logics of link type.

Transparallel literary and viewing structures may represent the future of tomorrow's literature. Hypermedia without transparallel viewing and transclusion is seriously limited, since it bars intercomparison and re-use. For tomorrow's electronic literature I believe we will need strong transclusion, full transparallelism, unrestricted republication and transcopyright.







# BIBLIOGRAPHY

1. T. Nelson, "A File Structure for the Complex, the Changing and the Indeterminate." Proceedings of the Association for Computing Machinery Twentieth Annual Conference, 1965.
2. T. Nelson, *Literary Machines 93.1*. Exists also in various earlier editions; those of 1985 and thereafter introduce the term "transclusion" and give the fullest treatment of Xanadu version 88.1. Mindful Press, 3020 Bridgeway #295, Sausalito, CA, USA 94965. (Now distributed by Eastgate Systems, Cambridge, Massachusetts.)
3. Xanadu Operating Company, "Xanadu Hypermedia Server Preliminary Documentation" (XOC DOC). Also available from Mindful Press.
4. T. Nelson, "Transcopyright: Pre-Permission for Virtual Republishing." Sapporo HyperLab, various dates and versions.
5. T. Nelson, "Retro-Hyperization with Xanalogical Structure: Issues in the Applicative Hyperization of Unwitting Systems." Presented at the ACM Workshop on "Adding Hypertext Functionality to Existing Systems," ACMHypertext 96 Conference, Washington, D.C., USA, March 1996. (Paper presented at the conference by Daniel Gross of Flow Research, Inc., New York, NY, USA.)  
  
(Presented as an Attachment to the present paper.)
6. T. Nelson, *Cinema of the Mind: Software Design Is the New Art of Virtuality*. Book in preparation.







## Appendix 0

# THE CLASSIC 1965 DESIGN

The details as given here are for the 1965 design. Modifications in the HyperLab implementation will be discussed later.

### EXACT DEFINITION OF ZIP STRUCTURE

The following is an axiomatic statement of 1965 zip structure. No attempt has been made to minimize it.

**There are two types of elements:** item and zips.

**Items.** An item is a text chunk or picture. Item units may not overlap, but must have sharp boundaries which are the same with respect to all connections.

\*/**\*** Allowing overlap among linked pieces is what makes the richer xanalogical systems “fine grain,” and what makes them much more difficult to implement.

**Zips.** A zip is a sequence of items.

**Connections.** All connections are between individual items. There are three types of connection: Sequence, Transclusion, Link. (Sequence connections may be thought of vertical, transclusion and link versions may be thought of as horizontal.) All structures, including zips, are created by individual connections of items to other items.

**Permitted vertical connections.** Vertical connections represent sequence. Each item may be connected vertically to at most two other items, above and below (its zipmates). If an item is not connected vertically, it is the only item on the zip.

**Permitted horizontal connections.** Horizontal connections represent linkage or shared material. Each item may be horizontally connected to one or zero items in any other zip. Such







a connection may be either a link or a transclusion. (If there are  $n$  zips, a given item may be connected to at most  $(n-1)$  other items, since it may be horizontally connected to at most one item in each other zip.)

There is no exact logical ground for this design. It represents an intuitive judgment as to simplicity, clarity and usability.\* Since the human mind is infinitely plastic and malleable, it can fit all ideas into this structure, or practically any other structure.

\*For this reason it will be important to be able to experiment with alternative structures at a later time.

Let me draw attention to the arbitrariness of the one-connection rule: that between any two zips, two elements can have at most one connection (link or transclusion). There is no exact logical ground for this rule either. It also represents an intuitive judgment as to simplicity, clarity and usability. (Note, however, that the one-connection rule places an upper limit on the complexity of the display.)





# Appendix 1

## COMMANDS OF THE HYPERLAB ZIP EDITOR

### THE OPERATIONS:

- New Zip
- New Item
- Save
- Open Existing Zip
- Hide Existing Zip
- Rearrange
- Copy
- Resize.
- Visual snapshot





## Appendix 2

# COMMONALITIES OF THE CLASSIC ZIP EDITOR and the proposed OPEN STANDARD FOR XANASTRUCTURE (OSX)

OSX, the Open Standard for Xanastructure (5), is a draft document to provide a standard xanalogical linking system to the computer community. Its distribution waits on permission from XOC, Inc. over certain proprietary issues.

Classical Zips (1) have certain interesting, if not remarkable, resemblances to OSX.

There are *virtual units of content* (“zips” in the zip editor, “documents” in Xanadu 88.1 and “works” in the XOC Server, “documents” in the proposed Open Standard for Xanadata.) These are virtual, consisting of lists; in order to assemble the unit, the system must go from the list to the operative storage to find the pieces.

There are *links*, which are specified in the data structure as elements connecting certain parts. These may be held in lists (the “butterfly trouser link” of the Open Standard for Xanadata and Xanadu 88.1) or tables (as in the HyperLab Zip Editor).

There are *transclusions*, which are not specifically noted in the data structure. Rather, the transclusions are found by *comparing the contents of different virtual units to find out what elements they have in common*. It will be readily seen that this is combinatorially demanding. However, other optimizations are possible.







## **APPENDIX 3: MAIN XANALOGICAL DESIGNS, 1960-95**

**including also some related non-xanalogical  
designs.**

1960. NELTRAN (Primordial Xanadu).

1965: Zipper Lists (design was called ELF, or Evolutionary List File).

1967. SNP (Sexus-Nexus-Plexus). Recognizing that this embodied functionality most properly put in the operating system, this was an operating-system-level design. The three modules defined an append-only applicative text space. To the best of my recollection, Sexus was the user interface, Nexus the portion which maintained the coupling structures, and Plexus maintained the storage and virtual objects.

1969 Xanadu: Scrolling linked windows implemented by parallel streams.

1971. Attempted implementation of a single-stream word processor (JOT) as a stepping stone to a multi-stream Xanadu system.

1972 Xanadu: Model T enfilade (still proprietary to XOC, Inc.).

1975. JABBERWOCK design: planned as a transparallel text system using multicouplers.

1976. Multicoupler system for proposed software of Itty Bitty Machine Co., Evanston IL. (Descended from JABBERWOCK.)

1973-9. William Barus' double enfilade (still proprietary to XOC, Inc.).

1981. VORTEXT system, designed at Datapoint Corporation. (With Mark Miller, Stuart Greene and Steve Witham). Parallel streams and pointers.

1981-current. ZigZag™ design: a token software system with unique interface and transparallel capability.





- 1979-88. Xanadu 88.1, designed by Mark Miller, Roger Gregory and Stuart Greene. (Described in some detail in *Literary Machines*).
- 1989-current. XOC server, designed by Mark Miller, Dean Tribble and Ravi Pandya, around central algorithm by K. Eric Drexler.
- 1996 proposal: Open Xanastructure Specifications. Intended to xanadize legacy texts, and to standardize form of link ("butterfly trouser link" over discontiguous spans) for communication among applications.
- 1996 proposal for Xanadu Enterprise Engine, creating a suite of text and audio functionality for individuals and small offices around Open Xanastructure Specifications.
- 1996 proposal for NewXanadu, combining ZigZag tokens and interface, fine-grain transclusive transparallel system, Netscape interaction, standard database.

THE END





## ATTACHMENT

Paper presented at  
Hypertext 96 Conference

March 1996

Washington, D.C.

***THE END***





# Hypertext 96 Conference





# ***RETRO-HYPERIZATION WITH XANALOGICAL STRUCTURE:***

## **ISSUES IN THE APPLICATIVE HYPERIZATION OF UNWITTING SYSTEMS**

Theodor Holm Nelson

For the Workshop on "Adding Hypertext Functionality to Existing Systems"  
Hypertext 96 Conference, Washington, D.C., March 1996.

(The paper was presented at the conference by Daniel Gross of Flow  
Research, Inc., New York, NY.)

Theodor Holm Nelson

Hokkaido University, Sapporo HyperLab, and Project Xanadu

(After 1 April: Keio University Shonan Fujisawa Campus,  
Fujisawa, Japan)

e-mail: ted@xanadu.net

Note that "Xanadu" is a registered trademark of the author.  
Proprietary search and data methods of the Xanadu project are the  
property of XOC Inc., licensed to Filoli, Inc.

-----  
The issue in this workshop is how unwitting systems are to be extended in  
hypertextual dimensions: the hyperization of external and legacy systems, and  
the retro-hyperization of ancient document bases and databases.

I will assume that the principal target systems are textual documents, though  
the following remarks may have applications elsewhere.

These remarks present a generalized model of the structure of the Xanadu®  
hypermedia system, suitable for use by anyone, and for the exchange of  
compound media data.

There are several issues in the retro-hyperization problem.





**1. The material must hold still after connections are made.** Addressing must be committed and stable. This involves various political issues: commitment, reliable long-term storage, and domains that will remain available.

## **2. Applicative links.**

Connections may be either APPLICATIVE, that is, outside the stored material on some other layer, or EMBEDDED (somehow implanted or concealed in the data, in the form of pointers, anchors, labels, etc.)

I strongly favor applicative methods, as they leave original materials untouched, as well as having a variety of other benefits.

I would like to propose a model of applicative links, which have always been a part of the Xanadu project. This particular form crystallized in Xanadu 88.1, discussed in my book *Literary Machines* 93.1 (1).

Links are APPLICATIVE and adhere to the ELEMENTS (typically text bytes) Thus links are to CONTENT and not to ANCHORS or to SLITS (addresses between characters).

This form of link may be visualized as a pair of trousers; thus the informal term "trouser link." The idea is that a collection of elements (esp. text bytes) on one side of the link is linked to a collection of elements on the other side of the link. These collections need not be consecutive; hence the "trouser" visualization.

Because of adherence to bytes, links persist through evolution. This permits tracking connections over CHANGE. Because links are to the elements themselves, their destiny in successive versions is well-defined: if any of the bytes remain, the link remains. (If the whole link is within one document, and only one side remains, it is in a stultified form, but gives access to the other side of the link in other versions.)

Note that all links have the same mechanism, whether they are created by the document owner or not; all users get to make "first class" links. This is distinct from many applications where users are allowed to make low-grade, inferior links of various kinds.







This is FAIR, as well as indefinitely extensible (unlike second-class links).

### **3. The xanalogical model: links and transclusions.**

In the xanalogical model (1), there are two forms of connections: LINKS, or connections between things which are different, and TRANSCLUSIONS, or connections among things which are THE SAME.\*

\* The term "xanalogical" has been used by the author, and other members of the Xanadu team, for this structure since approximately 1985. One malicious and dishonest account of Xanadu history has treated our design as "bragging" and a misunderstanding of how computers work, rather than what it is, a principled design which can be achieved in many ways.

Xanalogical structure, or xanastructure (neither claimed as a trademark) is hyperstructure allowing both links and transclusions. Links and transclusion are orthogonal relationships.

I continue to argue that the transclusion is the vital complement to the link. It is the way that things can be in more than one place, and compared in those different locations.

Transclusion may be understood either as ONE THING INSTANCED IN MANY PLACES, CROSS-VISIBLE (the conceptual level), or MANY COPIES MAINTAINING CONNECTION AND VIRTUAL IDENTITY (working back from the fact that implementation may be in any form). These are logically equivalent.

Transclusion permits portions to be freely re-used; transclusion permits links to continue to adhere; transclusion permits study of the same material in a variety of contexts under a variety of auspices; transclusion permits a pro-rata sale of material, potentially cleaning up a considerable subsector of the copyright problem.

### **4. CHANGE.**

Changes may be tracked by different methods.\*





\* (Some clever ones having been proprietary to the Xanadu project.)

The simplest is a Delta list, listing all the edits in a reversible form. (Deletion is represented as a rearrangement to a negative address.)

**5. THIS IS THE SIMPLEST MODEL.** It eliminates, conceptually, both files and directories. There are only versions, elements, links, and changes through hypertime. (A document is a nominal grouping of versions with an owned boundary, though not necessarily with owned data elements.)

## **6. NO "APPLICATIONS."**

The computer model of "applications" is built around the political issue of hoarding proprietary data structures. Using this data model as a common structure, "applications" are simply different viewers and tools.

## **7. MECHANICS, and THE EFFICIENCY PROBLEM.**

Alas, there is a drawback, and that is that the mechanics of link and transclusion search must be done by address comparisons, comparing lists for common spans.

This is a combinatorially vexing, not to say hideous, problem.

Our proprietary methods for sequential machines, still unfortunately secret, may be the best for sequential processors; but tomorrow's massively parallel processors probably will handle it with much greater ease.

The NewXanadu96™ will be this same model.

We are eager to see (and possibly license) the approaches to be taken by others.

## **BIBLIOGRAPHY**

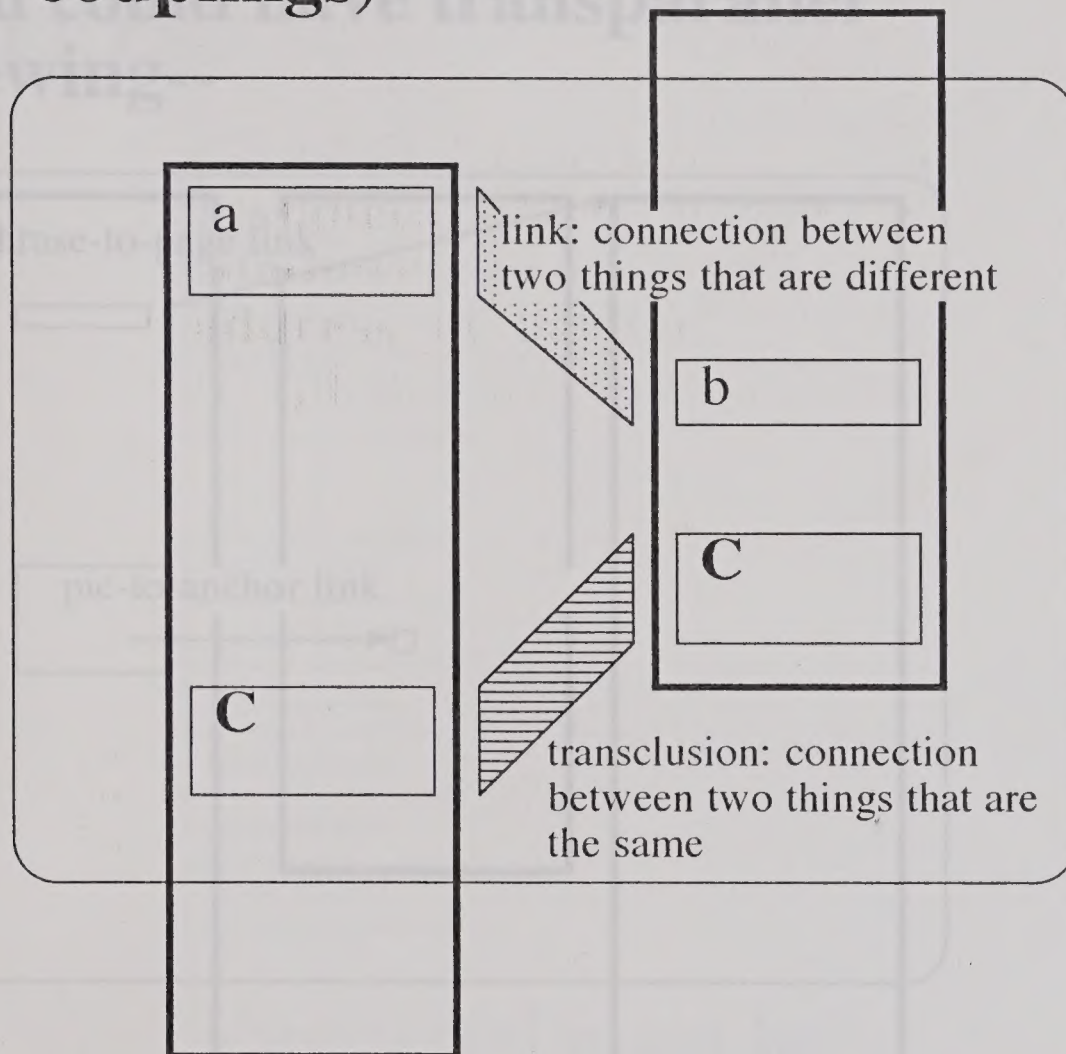
1. Theodor Holm Nelson, Literary Machines 93.1. Mindful Press, Sausalito, California; now being distributed by Eastgate Systems, Cambridge, Massachusetts.





*A long-term key goal of Xanadu®--*

## **TRANSPARALLEL VIEWING** (objects shown with detailed couplings)



**-- supported by no current browsers**

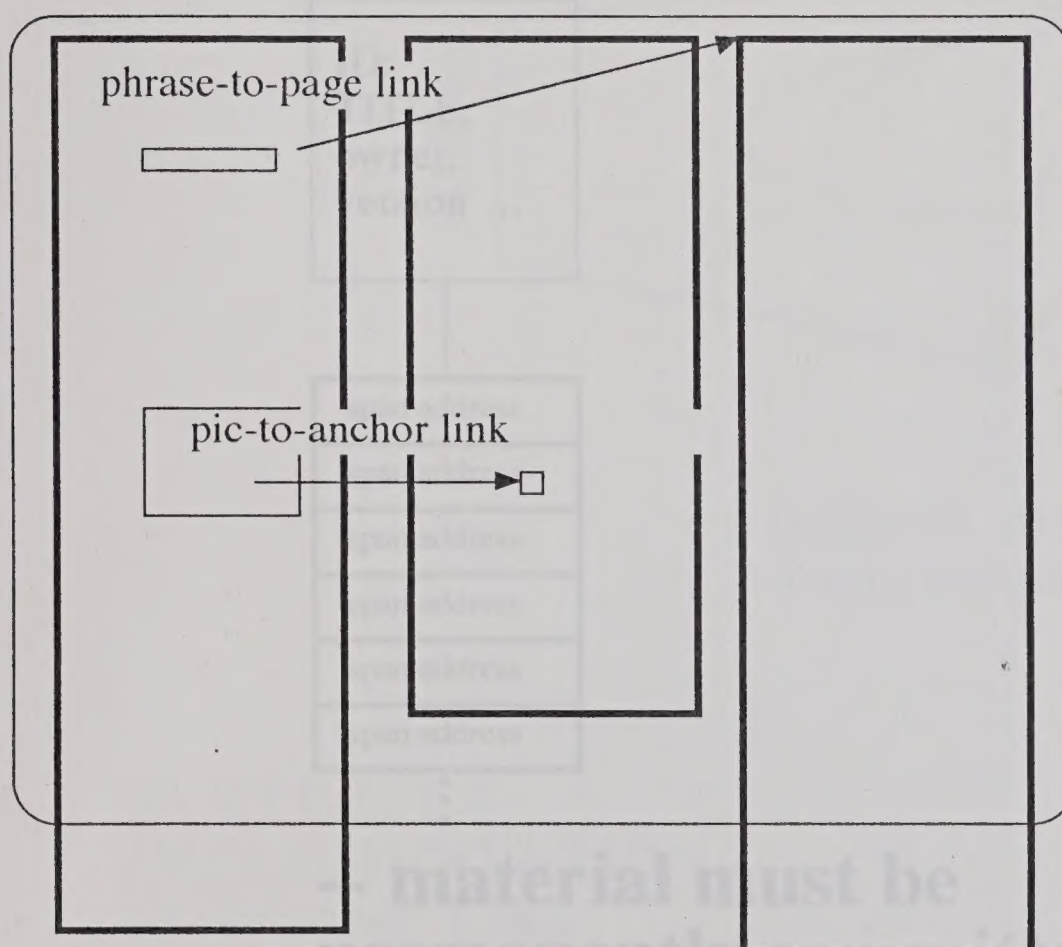
### **TRANSCCLUSION:**

Instances which may be re-used and compared in different contexts; keyed to address of master.  
(Implementation may vary)





**EVEN WITH TODAY'S WEB,  
you could have transparallel  
viewing--**



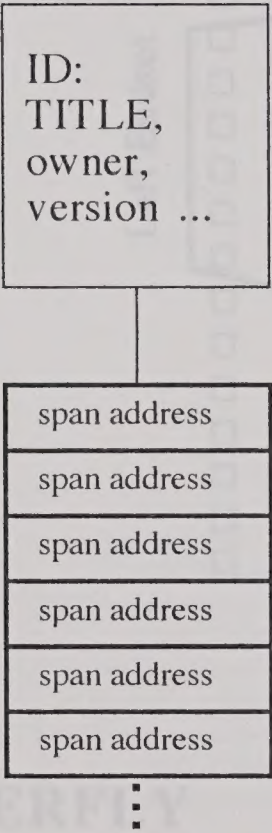
**-- if they cared to put it in**





XANASTRUCTURE LINKS  
("Butterfly Trouser Links")  
ARE ALSO LISTS OF PIECES.

**DOCUMENT (object, Work)  
REPRESENTED BY LIST OF  
PIECES.**



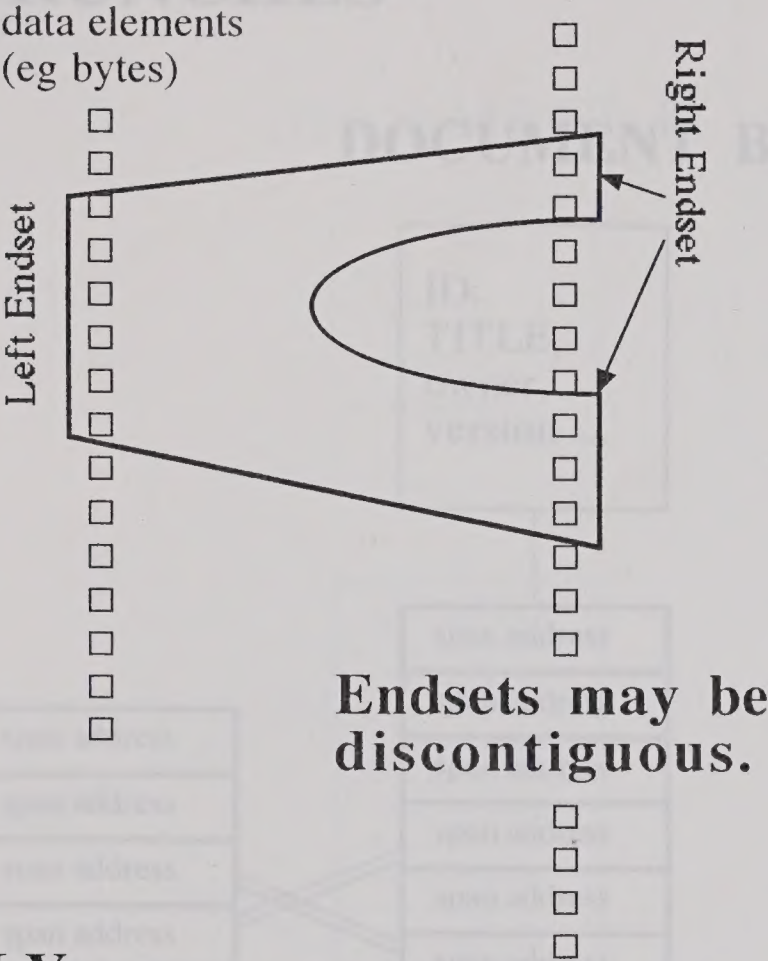
-- material must be  
permanently committed  
to storage



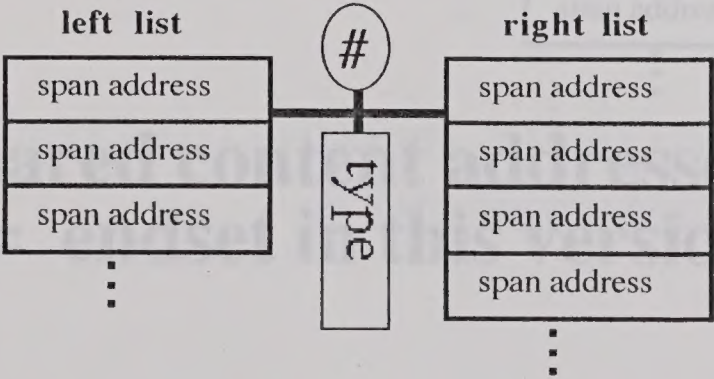


# XANASTRUCTURE LINKS ("Butterfly Trouser Links") ARE ALSO LISTS OF PIECES.

## Concept of TROUSER LINK



## BUTTERFLY IMPLEMENTATION of Trouser Link



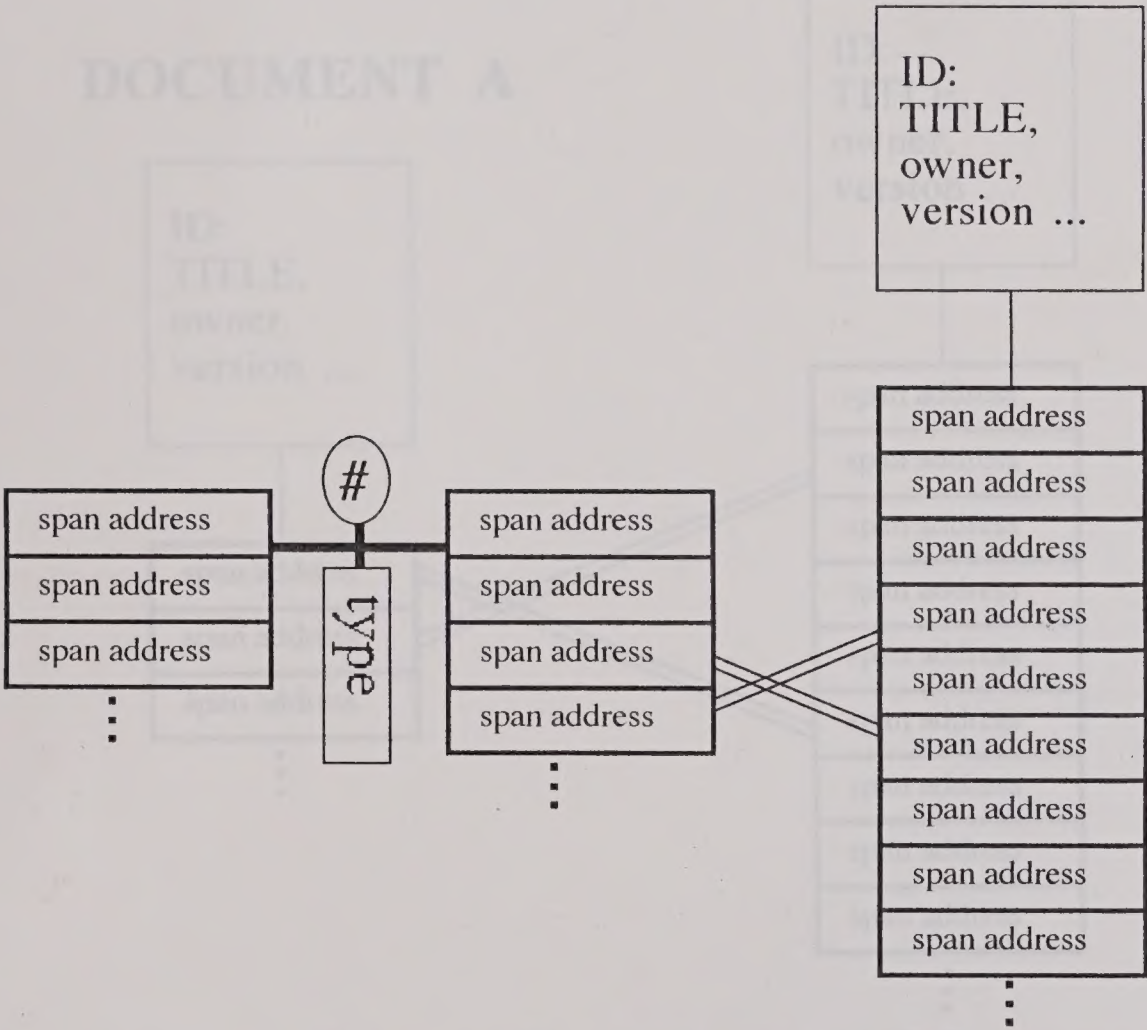




ADDRESS CRUNCH OPERATIONS  
follow links and find transclusions.

LINKS FOLLOWED BY  
ADDRESS CRUNCHES--

DOCUMENT B



Shared content addresses  
= endset in this version



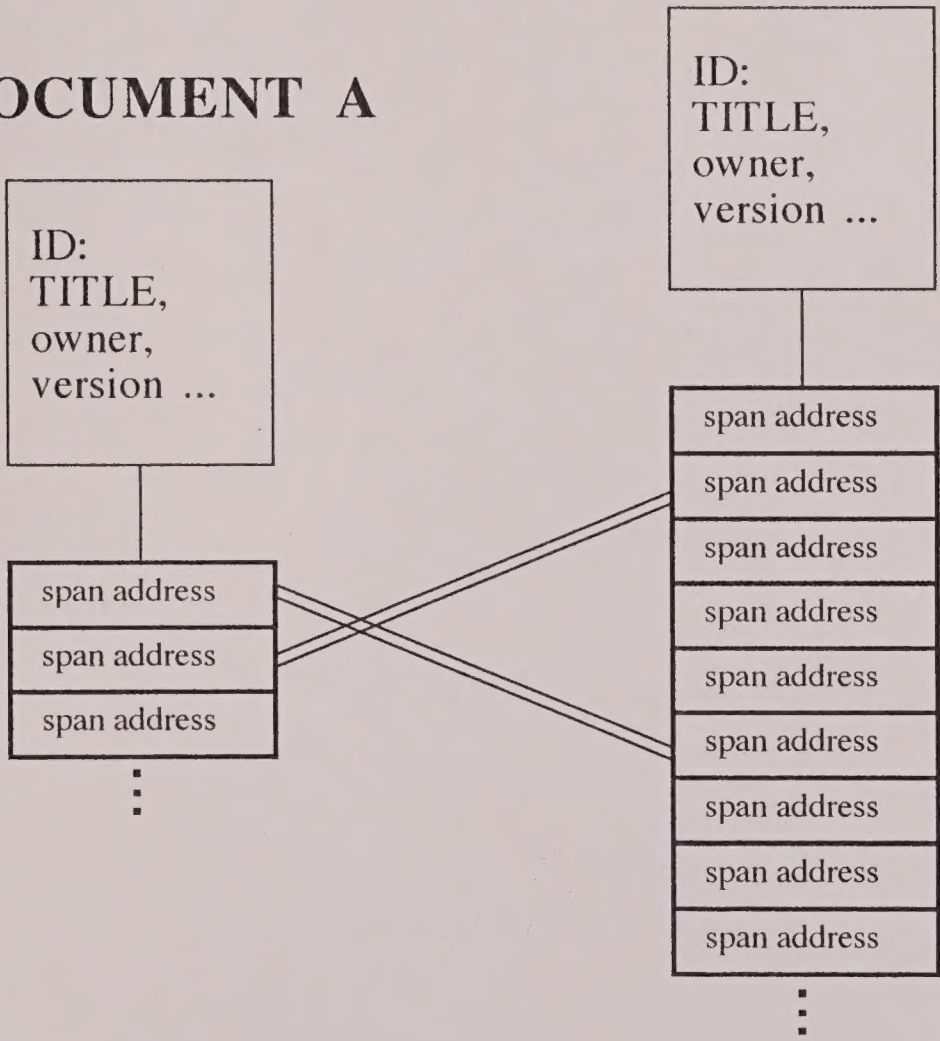


*ADDRESS CRUNCH OPERATIONS  
follow links and find transclusions.*

**TRANSCLUSSIONS  
FOUND BY ADDRESS  
CRUNCHES--**

**DOCUMENT B**

**DOCUMENT A**



**Shared content addresses  
= transclusions**







